SIMULTANEOUS COOPERATIVE EXPLORATION AND NETWORKING

A Thesis
Presented to
The Academic Faculty

by

Jonghoek Kim

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
May 2011

SIMULTANEOUS COOPERATIVE EXPLORATION AND NETWORKING

Approved by:

Dr. Fumin Zhang, Dr. Magnus Egerstedt, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Chuanyi Ji
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Patricio Vela
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Spyros Reveliotis
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Yorai Wardi
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date Approved: 16 March 2011

ACKNOWLEDGEMENTS

First of all, I appreciate my advisors, Dr. Fumin Zhang and Dr. Magnus Egerstedt, for helping me solve challenging and interesting research problems during my Ph.D. program. Their endeavor and support enabled the accomplishment of this dissertation. I also wish to thank Dr. Patricio Vela, Dr. Yorai Wardi, Dr. Chuanyi Ji, and Dr. Spyros Reveliotis for serving on my dissertation committee.

While pursuing Ph.D. degree, I enjoyed collaborating with outstanding people. I want to thank Sean Maxon for assisting me to do experiments related to this thesis. Also, I thank lab members of both Dr. Fumin Zhang and Dr. Magnus Egerstedt for their friendship and meaningful conversations: Waseem Abbas, Dongsik Chang, Rahul Chipalkatty, Jean-Pierre de la Croix, Greg Droge, Brandon Groff, Dr. Musad Haque, Hassan Jaleel, Dr. Hiroaki Kawashima, Peter Kingston, Amy LaViers, Shayok Mukhopadhyay, Dr. Amir Rahmani, Justin Shapiro, Zhenwu Shi, Klimka Szwaykowska, Philip Twu, Chuanfeng Wang, and Wencen Wu (sorted by last name).

I appreciate my family, who gave me endless love and unwavering support. Lastly, I thank God for guiding me to obtain Ph.D. degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

This thesis provides strategies for multiple vehicles to explore unknown environments in a cooperative and systematic manner. These strategies are called Simultaneous Cooperative Exploration and Networking (SCENT) strategies. As the basis for development of SCENT strategies, we first tackle the motion control and planning for one vehicle with range sensors. In particular, we develop the curve-tracking controllers for autonomous vehicles with rigidly mounted range sensors, and a provably complete exploration strategy is proposed so that one vehicle with range sensors builds a topological map of an environment. The SCENT algorithms introduced in this thesis extend the exploration strategy for one vehicle to multiple vehicles.

The enabling idea of the SCENT algorithms is to construct a topological map of the environment, which is considered completely explored if the map corresponds to a complete Voronoi diagram of the environment. To achieve this, each vehicle explores its local area by incrementally expanding the already visited areas of the environment. At the same time, every vehicle deploys communication devices at selected locations and, as a result, a communication network is created concurrently with a topological map. This additional network allows the vehicles to share information in a distributed manner resulting in an efficient exploration of the workspace.

The efficiency of the proposed SCENT algorithms is verified through theoretical investigations as well as experiments using mobile robots. Moreover, the resulting networks and the topological maps are used to solve coordinated multi-robot tasks, such as capturing intruders.

CHAPTER I

INTRODUCTION AND BACKGROUND

1.1 Introduction

As autonomous exploration and mapping become increasingly robust on a single vehicle [84, 32, 3], the next challenge is to explore unknown environments using multiple vehicles. Compared to exploration using a single vehicle, the extension to multiple vehicles poses a new challenge [44, 39, 11]. Coordination of multiple vehicles typically relies on communication between vehicles. But, direct communication is easily blocked or at least attenuated by obstacles. Recently, small sensing devices with relatively short communication range, such as the Berkeley MOTES[1], have become commercially available. A large number of such devices can be deployed to form a sensor network [24]. A device with sensing and communication capabilities can be viewed as an *information node*. Deployed information nodes form an *information network* that relays data across the network.

The developments of information nodes have inspired us to propose strategies called Simultaneous Cooperative Exploration and NeTworking (SCENT) strategies [53]. The SCENT algorithms proposed in this thesis construct the Voronoi diagram[2] as a topological map of a workspace. The workspace is considered completely explored if all Voronoi edges are explored. The vehicles are initially deployed at arbitrary locations and are not necessarily aware of the existence of other vehicles. Each vehicle explores its local area by incrementally expanding the already visited parts of the

[1] A Berkeley MOTE is a wireless sensor module manufactured by Berkeley. Typically, a sensor node is composed of: sensing capabilities, communication radio, computation unit, and a power source.

[2] Voronoi diagrams have been widely used for topological maps in robotics, c.f. [18, 73, 76, 15, 61], as well as for studying coverage problems in sensor networks [23, 68].

environment.

Every vehicle deploys information nodes at selected locations while constructing the Voronoi diagram of the explored workspace. As each vehicle builds up an information network, the networks built by different vehicles will eventually meet, allowing for inter-vehicle information sharing. This distinguishes the SCENT algorithms from other approaches [10, 44, 39, 11] that only allow robot-to-robot communication.

The SCENT algorithms proposed in this thesis are provably complete under mild technical assumptions. A performance analysis of the SCENT algorithms verifies that in a bounded workspace, the time spent to complete the exploration decreases as the number of vehicles increases. Analytical formulas for this relationship are provided in this thesis. Furthermore, simulation and experimental results are presented to demonstrate the effectiveness of the SCENT algorithms.

As a result of the SCENT algorithms, an information network is created concurrently with a topological map of the workspace. The resulting information network and the topological map can then be used to solve coordinated multi-robot tasks. We utilize the constructed network as a basis for capturing intruders in the workspace.

We study intruder capturing game on the topological map of the workspace, represented by the Voronoi diagram. We assume that a searcher can access the position of any intruder using the information network. Obeying the conventions established in the literature on graph searching problem [75, 70, 59, 60, 36, 4, 37, 67], an intruder can maneuver at unbounded speed to avoid searchers. Furthermore, an intruder has full knowledge of the environment, positions of the searchers, and the strategies of the searchers. An intruder is *captured* if it is forced to share a node with any searcher. This thesis provides an upper bound for the minimum number of searchers required to capture all intruders on a general graph, which leads to a result on the Voronoi diagram.

The organization of this thesis is as follows: the remainder of this chapter introduces the background research for the thesis. As the basis for development of the SCENT algorithms, we first tackle the motion control and planning for one vehicle with range sensors. In particular, we develop the curve-tracking controllers for autonomous vehicles with rigidly mounted range sensors (Chapter 2), and a provably complete exploration strategy is proposed so that one vehicle with range sensors builds a topological map of an environment (Chapter 3). The SCENT algorithms introduced in Chapter 4 extend the exploration strategy for one vehicle to multiple vehicles. Using the SCENT algorithms, multiple vehicles explore an unknown environment while deploying information nodes at selected locations. As a result, an information network is created concurrently with a topological map. Chapter 4 further shows that the constructed network can be used as a basis for capturing intruders in the workspace. Finally, we provide conclusions and future directions in Chapter 5.

1.2 Background Research

In this thesis, various questions from the field of autonomous robotics are investigated. This thesis tackles curve-tracking control (Chapter 2), construction of Voronoi diagrams (Chapter 3), multi-robot exploration and mapping (Chapter 4), and capturing intruders on graphs (Chapter 4). This section provides a background introduction to the entire thesis. We discuss the various previous works for curve-tracking control, Voronoi diagrams, multi-robot exploration and mapping, and capturing intruders on graphs.

1.2.1 Curve-Tracking Control

Curve-tracking control is fundamental for an autonomous vehicle following a desired path, e.g., staying in lanes or tracking obstacle boundaries. There are many papers on curve tracking for autonomous vehicles. For example, the authors of [62] determined the bounds for the sampling intervals in order to ensure that the vehicle stays in the

lane with a limited sensing rate. A biologically plausible feedback law that achieves motion camouflage, which is related to curve tracking, was shown in [46]. Curve tracking for an atomic force microscope was considered in [1].

Suppose that a robot is given a reference path, i.e., the robot is aware of the coordinates for all points on the path. In [35], a "virtual" vehicle approach was proposed to make a robot follow a reference path. The time evolution of the reference point, denoted as the *virtual vehicle*, was governed by a differential equation containing error feedback. If both the tracking errors and disturbances are within certain bounds, the virtual vehicle moves along the reference trajectory while the real robot follows it. Otherwise, the virtual vehicle slows down and waits for the real robot.

To make a robot track a reference path with a constant speed, a feedback linearization approach and Lyapunov-oriented control designs were presented in [71]. Since the robot is designed to maintain constant speed, this approach can be appropriate for vehicles that must maintain high speed (unmanned aerial vehicles). Curve-tracking control was developed for unicycle-type robots, as well as for two-steering-wheel mobile robots[3].

In a real application scenario, there may be the case where the robot is not given a reference path. In this case, the robot has to track a curve based on sensor measurements, such as vision sensors or range sensors.

Various vision-based path following methods were discussed in [22, 28, 29, 77]. The authors of [66] developed control laws so that a nonholonomic mobile robot equipped with vision sensors tracks an arbitrarily shaped continuous ground curve. This tracking problem was formulated as controlling the shape of the curve in the image plane. The authors studied the controllability of the system characterizing the dynamics of the image curve. In addition, they presented stabilizing control

[3]Two-steering-wheels mobile robot can be regarded as a unicycle-type robot with one additional degree of freedom which allows the orientation of the robot to be controlled independently of the path's direction.

laws for tracking piecewise analytic curves and proposed tracking arbitrary curves by approximating them as piecewise linear curvature curves. Here, *linear curvature curve* is a curve satisfying that the derivative of its curvature $k(s)$ with respect to the arc length parameter, s, is a nonzero constant, i.e., $\frac{\partial k(s)}{\partial s} = C \neq 0$.

A robot can track a boundary curve using range sensor measurements. We can assume that the range sensors of the robot have the ability to determine a point on an obstacle boundary that is closest to the robot. This point is called the *closest point*. In [89], a gyroscopic feedback law was used to control the model that describes the interaction between the robot and an image particle representing the closest point on an obstacle boundary. This controller design method was extended to set up cooperative motion patterns for multiple robots [91, 87, 92] and was generalized to the design of tracking laws in three dimensions [78, 45]. The closest point was also used for path following in [80].

To gather information of the closest point, the robot must be equipped with wide-aperture (up to 360 degrees) scanning sensors. Chapter 2 introduces curve-tracking control which only requires two narrow aperture range sensors, pointing to a fixed direction relative to the heading direction of the robot. Under such a sensor configuration, singularities are bound to occur in the control laws when tracking concave curves. In order to overcome these singularities, we derive a hybrid strategy of switching between control laws when the robot gets close to singularities.

1.2.2 Voronoi Diagrams

Voronoi diagrams are named after Georgy Fedoseevich Voronoi, who defined Voronoi diagrams in 1908. In 1965, Brown [9] studied the intensity of trees in a forest. He defined the area potentially available to a tree as the Voronoi cell of that tree. One year later, Mead [69] used the same concept for plants, referring to Voronoi cells

as *plant polygons*. Recently, there has been an impressive amount of literature regarding Voronoi diagrams and their applications in various research areas, such as computational geometry [38, 57, 58] and robotics [18, 19, 73, 76].

We review various applications of Voronoi diagrams in robotics. In robotics, Voronoi diagrams have been widely used for studying coverage problems in sensor networks [23, 68], as well as for topological maps of an environment [18, 73, 76, 15, 61]. Voronoi diagrams can be generalized into higher dimensions and can fit a wide class of robots with higher dimensional configuration spaces [14, 15].

Voronoi diagrams have been utilized to perform coverage optimization using multiple sensor platforms [23]. Given a density function that describes the probability of events happening and a performance function that measures the cost to service a location, the authors of [23] considered the problem of positioning mobile sensors in the environment so as to minimize the expected servicing cost. To monotonically minimize the servicing cost, each mobile sensor is steered toward the centroid of the Voronoi partition generated by the sensor.

In robotics, Voronoi diagrams have been used for path planning [76, 5]. The visibility graph is a well-known data structure for computing the shortest collision-free path between a start and goal configuration. However, the shortest path is, in general, tangential to an obstacle, so a path computed from the visibility graph does not have any clearance. On the other hand, planning motion paths using the Voronoi diagram of the obstacles yields a path with maximal clearance; however, this path may be significantly longer than the shortest path. Hence, the authors of [86] introduced the VV(c)-diagram (the Visibility-Voronoi diagram for clearance c) which is a hybrid of these two approaches. The VV(c)-diagram evolves from the visibility graph to the Voronoi diagram as c grows from 0 to ∞, where c is the preferred amount of clearance.

In addition, Voronoi diagrams can be used for topological maps of an obstacle

environment. Several extensions of Voronoi diagrams were developed by robotic researchers, including the generalized Voronoi graph (GVG), hierarchical GVG (HGVG), and the reduced GVG (RGVG)[4] in [18, 19, 73]. The GVG, HGVG, and the RGVG were developed to be used as a *roadmap* [17] of a robot. The roadmap is a concept having the following properties: *accessibility, connectivity, and departability.* These properties imply that the robot can construct a path between any two points in a connected component of the robot's free space. For example, suppose that the robot builds a path from one point A to another point B. Initially, the robot finds a path from A to the roadmap (accessibility). Then, it traverses the roadmap to the vicinity of B (connectivity) and constructs a path from the point on the roadmap to B (departability). Section 1.2.2.1 reviews incremental construction of the GVG in detail.

1.2.2.1 Incremental Construction of the Generalized Voronoi Graph (GVG)

This section reviews incremental construction of the generalized Voronoi graph (GVG) in [20].

Consider a connected and compact workspace $W \subset \mathcal{R}^2$ whose boundary, ∂W, is a regular curve. Let O_1, O_2,..., O_n be n closed and convex obstacles in W. Non-convex obstacles are modeled as the union of convex obstacles. The GVG is based on the following distance function:

$$d_i(x) = min_{c_0 \in O_i} \|x - c_0\|, \quad (1)$$

$$\nabla d_i(x) = \frac{x - c_0}{\|x - c_0\|}, \quad (2)$$

where $d_i(x)$ is the distance from a point x to an obstacle O_i, and the vector $\nabla d_i(x)$ is a unit vector in the direction from c_0 to x, where c_0 is the closest point to x in O_i.

[4]The HGVG connects the GVG when the GVG is disconnected [19]. The reduced GVG gives better localization than the GVG, since no Voronoi edge is connected to the obstacle boundaries [73].

If a robot is located at x, then $d_i(x)$ and $\nabla d_i(x)$ can be computed using the range sensors of the robot.

The basic building block of the GVG is the set of points equidistant from two obstacles O_i and O_j such that each point in this set is closer to O_i and O_j than to any other obstacles. This structure is termed the *generalized Voronoi edge*,

$$F_{ij} = \{x \in \mathcal{R}^2 : 0 \leq d_i(x) = d_j(x) \leq d_h(x) \ ,\forall h \neq i,j \text{ and } \nabla d_i(x) \neq \nabla d_j(x)\}. \quad (3)$$

Observe that F_{ij}, F_{ik}, and F_{jk} intersect to build a structure that is equidistant from three obstacles. Such a structure is termed a *meet point* and is denoted as F_{ijk}:

$$F_{ijk} = F_{ij} \bigcap F_{ik} \bigcap F_{jk}. \quad (4)$$

Also, when a GVG edge intersects the obstacle boundary at a point, this point is termed a *boundary point*. A meet point and a boundary point are depicted in Figure 1.

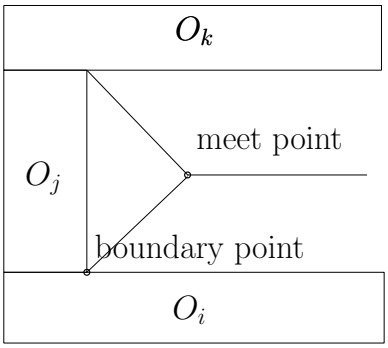


Figure 1: A meet point and a boundary point.

The robot accesses a GVG edge by following the gradient of the distance function to the nearest obstacle. See Figure 2. An edge-tracing method is used so that the robot on a GVG edge traces the GVG edge using range sensors.

Suppose that the robot is located at a point x on a GVG edge. Choose local coordinates at x so that the first coordinate, z_1, lies tangential to the GVG edge at x (see Figure 3). At x, let the plane normal to z_1 be termed the *normal slice*. The local

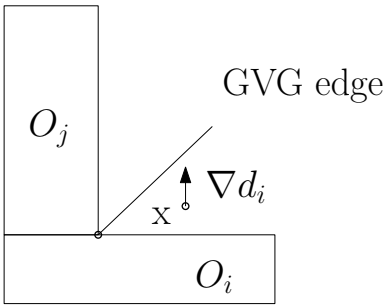


Figure 2: The robot at x following gradient of the distance function to the nearest obstacle.

coordinates are decomposed into (y, λ), where $\lambda = z_1$ is termed the *sweep coordinate*, and $y = z_2$ is the *slice coordinate*.

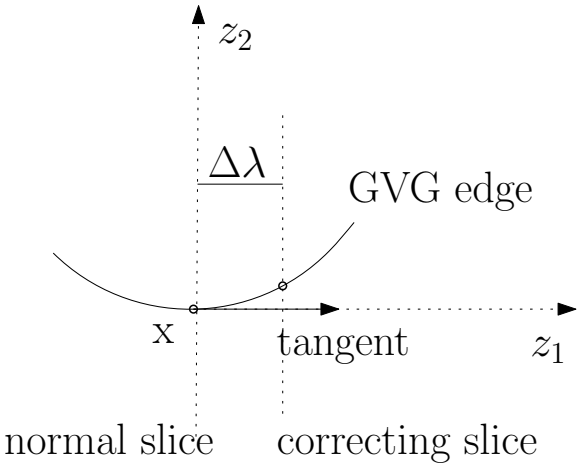


Figure 3: A normal slice and a correcting slice.

A robot traces a GVG edge using an adaptation of a predictor-corrector scheme. This scheme consists of two iterative stages: a prediction step, and a correction procedure. In a prediction step, the robot takes a small step, $\Delta\lambda$, in the z_1-direction. Usually, this prediction step will make the robot move off the GVG edge. Next, a correction procedure is used to bring the robot back onto the GVG edge. If $\Delta\lambda$ is small, then the GVG edge will intersect a *correcting slice* (Figure 3), which is a slice parallel to the normal slice at distance $\Delta\lambda$. The correction procedure finds the coordinate where the GVG edge intersects the correcting slice (Figure 3) using an iterative Newton's method. Then, the robot moves toward the intersecting coordinate.

After the correction procedure, the robot must rotate again to re-orient itself on a Voronoi edge. These rotations take time and cause additional wheel slippage [16]. Furthermore, since the prediction step makes the robot move off the GVG edge, this method may produce a zigzag path of the robot [16][5].

We review the exploration algorithms in [20] so that the robot visits all Voronoi edges in the workspace. Before stating the algorithms, the following concepts need to be introduced. *New meet point* denotes a meet point that the robot visits for the first time. In addition, *old meet point* denotes a meet point that is not a new meet point. The robot incrementally traces a GVG edge until it encounters a new meet point, an old meet point, or a boundary point.

When the robot encounters a new meet point, it marks off the direction from which it came as explored and identifies all unvisited GVG edges that emanate from it. From the meet point, the robot traces an unvisited GVG edge until it detects either another meet point or a boundary point. In the case where the robot detects another new meet point, the robot repeats the process in this paragraph.

When the robot reaches an old meet point, it has completed a cycle in the GVG graph. In this case, the robot travels to a meet point with an unvisited GVG edge followed by traversing the unvisited edge. When the robot reaches a boundary point, it simply traces back and returns to a meet point with an unvisited GVG edge followed by traversing the unvisited edge.

The exploration algorithms end when all meet points have no unvisited GVG edges. Note that convergence of the exploration algorithms has not been proven. Although many results exist in the literature for constructing Voronoi diagrams, to our knowledge, the boundary expansion algorithms in Chapter 3 are unique, with provable completeness over a a connected and compact workspace.

[5]In Chapter 3, we introduce a Voronoi edge tracking control law which guarantees that curvature is well-defined along the trajectory of a robot. This further implies that the trajectory of the robot is smooth.

1.2.3 Multi-Robot Exploration and Mapping

As autonomous exploration and mapping become increasingly robust on a single robot [84, 32, 3], the next challenge is to explore unknown environments using multiple robots. In this subsection, we review previous works on multi-robot exploration and mapping.

The problem of exploring an unknown environment using a team of robots was considered in [10, 11] under the assumption that the robots know their relative starting positions. This assumption is limiting in practice: it implies either that all robots start from the same location, or that the initial locations of the robots have been surveyed before the exploration begins. The goal of [10, 11] was to minimize the overall exploration time using multiple robots.

We briefly review the exploration and mapping strategy presented in [11]. For time-efficient exploration, the robots must keep track of which areas of the environment have already been explored. Thus, the robots construct a shared map and coordinate their actions using this map. The individual robots choose appropriate target points so that they simultaneously explore different regions of their environment. Define the *utility of target points* as the size of the unexplored area that a robot can cover with its sensors upon reaching a target position. A probabilistic approach is used for the coordination of multiple robots, taking into account both the costs of reaching a target point and the utility of target points. Whenever a target point is assigned to a specific robot, the utility of the unexplored area visible from this target position is reduced for the other robots. In this way, the individual robots are assigned to different target points.

Multi-robot exploration and mapping under uncertainty about the robots' relative starting locations are considered in [44]. This corresponds to the situation where the robots are placed at widely separated initial locations in unknown environments. Since the robots' relative starting locations are unknown, the robots determine their

relative pose only after pairs of robots encounter each other. The algorithm in [44] is based on the particle filter. Suppose two robots encounter each other at some time during the exploration. At this time, they determine their relative pose and the particle filter is initialized using the measured relative pose. Subsequent measurements from the two robots are fed to the filter, and thereby fused into shared maps. At the same time, two "virtual" robots are added to the filter so that data recorded before the encounter are incrementally fused into shared maps. Previously recorded measurements are fed to the filter in reverse time-order, such that these "virtual" robots appear to be driving backwards through the environment.

The approach in [44] had one crucial limitation: the state space in the particle filter was extremely large (with hundreds or thousands of dimensions), while the number of particles was necessarily small (a few hundred to a few thousand at most). Thus, the filter was a sparse sampling of the state space. To resolve this under-sampling problem, relatively accurate pose estimates were required. Thus, the author of [44] considered a robot equipped with scanning laser range-finders and combined odometry with laser scanning data to improve the accuracy in odometric pose estimates.

The authors of [39] also tackled multi-robot exploration and mapping under uncertainty about the robots' relative starting locations. The algorithm in [39] was based on the particle filter. In [39], the authors acknowledged that their approach did not scale to large teams of robots, since the particle filter was computationally expensive. They mentioned that, in the worst case, the number of particle filters run on each robot was as high as the total number of robots minus one. In the simulation results, the authors of [39] found that their approach worked efficiently for up to six robots.

Since the robots' relative starting locations are unknown in [44, 39], pairs of robots must encounter each other so as to merge their maps. However, one cannot guarantee that pairs of robots meet each other during the exploration process. Furthermore,

the authors of [44, 39] did not present the relationship between the time spent to complete the exploration and the number of robots deployed. Note that the only result of the exploration was to build the maps measuring an obstacle environment.

In Chapter 4, we provide the SCENT algorithms to explore an unknown environment using multiple robots under uncertainty about the robots' relative starting locations. A performance analysis of the SCENT algorithms shows that in a bounded workspace, the time spent to complete the exploration decreases as the number of robots increases. We provide an analytical formula for this relationship. Furthermore, as a result of the SCENT algorithms, an information network is created concurrently with a topological map of the workspace. The resulting information network and the topological map can be used to solve coordinated multi-robot tasks. Chapter 4 shows that the constructed network can be used as a basis for capturing intruders in the workspace.

1.2.4 Capturing Intruders on Graphs

Many papers exist on capturing intruders on graphs [70, 59, 60, 36, 37]. The author of [75] defined the graph searching problem. In this problem, searchers and intruders move along edges of a graph. An intruder can maneuver at unbounded speed to avoid searchers. Furthermore, an intruder has full knowledge of the environment, positions of the searchers, and the strategies of the searchers. An intruder is *captured* if it shares a vertex with a searcher and the intruder cannot make any move to escape. The *search number* $S(G)$ of a graph G is the minimum number of searchers needed to capture all intruders in G [67, 40, 37, 75].

An edge is *cleared* if every intruder on the edge is captured. Even if an edge is cleared, another intruder may enter the edge later. This situation is called the *recontamination* of an edge [60]. A search plan that does not involve recontamination of any edges is called a *monotone searching strategy*. The authors of [60] proved

that recontamination does not decrease $S(G)$, i.e., there always exists a monotone searching strategy with $S(G)$ searchers.

Based on the result that recontamination does not decrease $S(G)$, the authors of [70] showed that determining whether $S(G) \leq k$ where k is a positive constant is NP-complete. They also provided a structural characterization of those graphs G with $S(G) \leq N$ for $N = 1, 2, 3$.

Moreover, based on the result that recontamination does not decrease $S(G)$, the authors of [67] derived the relationship between $S(G)$ and the cutwidth of G. Before presenting the definition of the cutwidth of G, we define a *linear layout* of G as a one-to-one function mapping the vertices of G to integers. The *cutwidth* of G, denoted as $cw(G)$, is the smallest integer k such that the vertices of G can be arranged in a linear layout $[V_1, ..., V_n]$ in such a way that, for every $i = 1, ..., n-1$, there are at most k edges with one end point in $\{V_1, ..., V_i\}$ and the other in $\{V_{i+1}, ..., V_n\}$. In [67], it was proven that $S(G)$ is related to the cutwidth of G as follows: $S(G) \leq cw(G) \leq \lfloor deg(G)/2 \rfloor \cdot S(G)$, where $deg(G)$ denotes the maximum number of edges incident to any vertex in G. In particular, this implies that, for any graph G with $deg(G)$=3, $S(G) = cw(G)$.

There are many variants of the graph searching problem originated from [75]. In *node search* [55, 56], the searchers cannot slide along edges and an edge is cleared when both its endpoints are occupied by searchers. Another variant is *mixed searching* introduced in [6]. In mixed searching, sliding is permitted and an edge can be cleared either by sliding or by simultaneous occupation of both its endpoints. In the original graph searching problem, an intruder is *invisible*, i.e., the searchers have no information on the intruder's current position. Furthermore, we can consider the case where the invisible intruder is *lazy*, in the sense that the intruder moves only when one searcher is planning to occupy the vertex where the intruder currently is. This invisible-but-lazy variant was defined in [27].

In the helicopter cops and robbers game [81, 79], an intruder is *visible*, since the cops have complete knowledge of robbers' positions as if the cops are using helicopters. Furthermore, the cops can be placed on or removed from vertices of the graph. A robber is captured when a cop lands on the vertex occupied by the robber and the robber cannot make any move to escape. It was found in [79] that if we only consider monotone searching strategies, then the minimum number of cops required depends on the number of robbers. Let $MS(G, r)$ denote the minimum number of cops required to capture r robbers using monotone searching strategies. The authors of [79] derived the relationship between $MS(G, r)$ and the proper pathwidth defined in [83]. Moreover, they obtained the relationship between $MS(G, r)$ and the proper treewidth defined in [25]. For a general graph G, it was proven in [79] that $MS(G, r) \leq min\{ppw(G), ptw(G) \cdot (\lfloor \log r \rfloor + 1)\}$, where $ppw(G)$ and $ptw(G)$ denote the proper pathwidth and proper treewidth of the graph G respectively.

In Chapter 4, we assume that a searcher utilizes the information network to detect intruders, similar to a cop using a helicopter in [81, 79]. However, in Chapter 4, a searcher moves along edges of a graph continuously, which is distinct from [81, 79] and is closer to autonomous robot applications. In addition, our searching strategy is not monotone, which implies that even if every intruder on an edge is captured, another intruder may enter the edge later. Based on this searching strategy, we derive theoretical upper bound for the minimum number of searchers required to capture all intruders on a general graph, which leads to a result on the Voronoi diagram. Note that this upper bound does not depend on the number of intruders.

CHAPTER II

CURVE-TRACKING CONTROL FOR AUTONOMOUS VEHICLES WITH RIGIDLY MOUNTED RANGE SENSORS

Curve-tracking control is fundamental for autonomous vehicles following desired paths, e.g. staying in lanes or tracking obstacle boundaries. An example in which this becomes relevant is when an autonomous vehicle is to follow curbs or lane markings. Figure 4 shows the autonomous vehicle Sting-I that represented Georgia Tech in the DARPA Urban Grand Challenge in 2007. As one of this vehicle's lane perception strategies, two rigidly mounted range sensors(lidars) were installed on both sides of the vehicle. At each instant of time, the vehicle emits a group of laser rays around the center ray forming a fixed angle with the heading direction of the vehicle. When the center ray intersects a lane, it detects a point on the lane. From the distance measurements taken by the rays around the center ray, the autonomous vehicle is able to estimate the curvature of the lane at the point, the distance from the point, and the angle between the heading vector of the vehicle and the tangent vector to the lane.

Figure 4: The Sting-1 vehicle at Georgia Tech.

In this chapter, we design a curve-tracking controller that uses these measurements as feedbacks to create the desired lane-following behavior to be used as a component in the Georgia Tech urban grand challenge system. It should be noted that our results can be applied to other types of autonomous vehicles with similar range sensor configurations.

The literature is abundant with papers on curve tracking for autonomous vehicles [35, 89, 65, 93, 91, 87, 88, 78, 80, 45, 41, 1]. In the literature listed above, curve-tracking control usually has difficulties when the curve is concave, i.e., curving towards the vehicle. In this chapter, we follow a similar procedure as in [89] to develop curve-tracking controllers for both convex and concave curves based on Lyapunov functions. However, our results are significantly different, and hence complementary to those in [89]. First, information of the closest point is used in [89], which requires wide aperture scanning sensors. The method in this chapter only requires two narrow aperture range sensors, each of which forms a fixed direction relative to the heading direction of the vehicle. This sensor configuration not only makes the tracking dynamics more complicated, but also causes singularities in control laws when tracking concave curves. We show that these singularities cannot be avoided by changing the shape of the Lyapunov function used in [89]. Therefore, to overcome singularities of the Lyapunov-based control laws, we develop switching controllers to make the system asymptotically stable. The switching strategy that achieves curve tracking with narrow aperture range sensors is our main contribution in this chapter.

The proof of the convergence for our switching controllers is inspired by convergence results for switching systems in the literature. Conditions for nonlinear switching systems to be asymptotically stable were presented in [43]. In [63], [7], and [26], multiple Lyapunov functions were used to prove stability. In [85], the authors proposed control laws that switch between an approximate control when the system is near a singularity, and an exact control when the system is bounded away from the

singularity.

This chapter is organized as follows: In Section 2.1, we present a system model for curve tracking with rigidly mounted range sensors. In Section 2.2, we select a Lyapunov function for the convergence analysis and derive a feedback control law to asymptotically stabilize this system. Furthermore, to avoid singularities of the feedback control law, switching control laws are developed. Simulation results are presented in Section 2.3. Section 2.4 provides conclusions.

2.1 Boundary-Following Model with Rigidly Mounted Range Sensors

Consider a vehicle that emits a center ray forming a fixed angle α with the heading direction of the vehicle. See Figure 5 for the illustration of a vehicle and a boundary curve. When a boundary curve is presented in the plane, the center ray will intersect the boundary and detect the point $\mathbf{r}_2$, which will be called the *detected point*. Let $\mathbf{r}_1$ denote the position of the vehicle. Then, the relative position between the vehicle and the detected point is $\mathbf{r}_\alpha = \mathbf{r}_2 - \mathbf{r}_1$. ϕ is the angle measured counterclockwise from $\mathbf{x}_2$, the tangent vector to the boundary curve at the detected point, to $\mathbf{x}_1$, the heading direction of the vehicle.

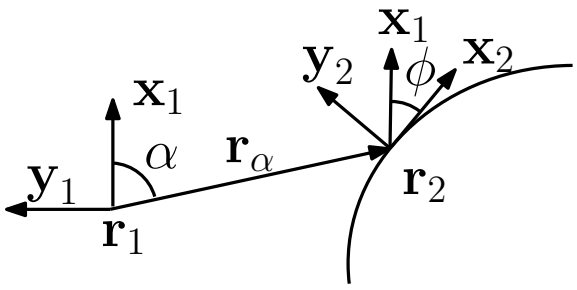


Figure 5: A vehicle emitting a center ray, which forms a fixed angle α with the heading direction of the vehicle.

The authors of [89] introduced Frenet-Serret frames [13] for the interaction between the vehicle and the boundary curve. Inspired by the frames in [89], we establish

two Frenet-Serret frames: one at the vehicle, and the other at the detected point. These two frames satisfy the Frenet-Serret equations:

$$\begin{aligned}\dot{\mathbf{r}}_1 &= v_1\mathbf{x}_1 \\ \dot{\mathbf{x}}_1 &= v_1 u\mathbf{y}_1 \\ \dot{\mathbf{y}}_1 &= -v_1 u\mathbf{x}_1 \end{aligned} \tag{5}$$

$$\begin{aligned}\dot{\mathbf{r}}_2 &= \dot{s}\mathbf{x}_2 \\ \dot{\mathbf{x}}_2 &= \dot{s}\kappa\mathbf{y}_2 \\ \dot{\mathbf{y}}_2 &= -\dot{s}\kappa\mathbf{x}_2, \end{aligned} \tag{6}$$

where v_1 is the speed control, and u is the steering (i.e., curvature) control we apply to avoid colliding with obstacles and to achieve boundary following. In addition, κ is the curvature of the curve at the detected point obtained using a group of rays around the center ray, and s is the arc length parameter of the curve. We choose the positive direction of the boundary curve such that

$$\mathbf{x}_1 \cdot \mathbf{x}_2 = \cos(\phi) > 0. \tag{7}$$

When the curve is convex, i.e., curving away from the vehicle, we have $\kappa < 0$. When the curve is concave, i.e., curving towards the vehicle, we have $\kappa > 0$.

The key idea of curve-tracking control is to control the relative motion between the vehicle and the detected point. For this purpose, we develop the set of equations governing the relative motion.

The relative position between the vehicle and the detected point is ($\mathbf{r}_\alpha = \mathbf{r}_2 - \mathbf{r}_1$). In Figure 5, α is defined as the angle formed by $\mathbf{r}_\alpha$ and $\mathbf{x}_1$. Also, let $r_\alpha = \|\mathbf{r}_\alpha\|$. Then

$$\mathbf{r}_\alpha \cdot \mathbf{x}_1 = \cos(\alpha) r_\alpha. \tag{8}$$

To derive the relative motion equations, we need to find $\dot{r}_\alpha$, $\dot{s}$, and $\dot{\phi}$.

We first obtain an equation linking $\dot{r}_\alpha$ with $\dot{s}$. Take the time derivative of $\mathbf{r}_\alpha$ using (5) and (6) to get

$$\dot{\mathbf{r}}_\alpha = \dot{s}\mathbf{x}_2 - v_1\mathbf{x}_1. \tag{9}$$

Differentiating (8) with respect to time on both sides, we obtain

$$\dot{\mathbf{r}}_\alpha \cdot \mathbf{x}_1 + \mathbf{r}_\alpha \cdot \dot{\mathbf{x}}_1 = \cos(\alpha)\dot{r}_\alpha. \tag{10}$$

And then, replacing $\dot{\mathbf{x}}_1$ by $v_1 u\mathbf{y}_1$, we get

$$\dot{\mathbf{r}}_\alpha \cdot \mathbf{x}_1 + \mathbf{r}_\alpha \cdot v_1 u\mathbf{y}_1 = \cos(\alpha)\dot{r}_\alpha. \tag{11}$$

Replacing $\dot{\mathbf{r}}_\alpha$ in (11) by (9), we obtain

$$(\dot{s}\mathbf{x}_2 - v_1\mathbf{x}_1) \cdot \mathbf{x}_1 + \mathbf{r}_\alpha \cdot v_1 u\mathbf{y}_1 = \cos(\alpha)\dot{r}_\alpha. \tag{12}$$

Observe that, in Figure 5, the angle formed by $\mathbf{x}_1$ and $\mathbf{x}_2$ is ϕ, and the angle formed by $\mathbf{r}_\alpha$ and $\mathbf{y}_1$ is $(\frac{\pi}{2} + \alpha)$. Therefore, $\mathbf{x}_1 \cdot \mathbf{x}_2 = \cos\phi$ and $\mathbf{r}_\alpha \cdot \mathbf{y}_1 = -r_\alpha \sin\alpha$. Applying these two equations to (12), we get

$$\dot{s}\cos(\phi) = v_1(1 + sin(\alpha)r_\alpha u) + \cos(\alpha)\dot{r}_\alpha. \tag{13}$$

Now, noticing that

$$r_\alpha^2 = \|\mathbf{r}_\alpha\|^2 = \mathbf{r}_\alpha \cdot \mathbf{r}_\alpha, \tag{14}$$

an equation linking $\dot{r}_\alpha$ with $\dot{s}$ can be established. We differentiate (14) with respect to time on both sides to obtain

$$2r_\alpha\dot{r}_\alpha = 2(\dot{s}\mathbf{r}_\alpha \cdot \mathbf{x}_2 - v_1\mathbf{r}_\alpha \cdot \mathbf{x}_1), \tag{15}$$

where we have used (9). Then, $\dot{r}_\alpha$ is

$$\dot{r}_\alpha = \dot{s}\frac{\mathbf{r}_\alpha}{r_\alpha} \cdot \mathbf{x}_2 - v_1\cos(\alpha), \tag{16}$$

where we used the fact that the angle formed by $\mathbf{r}_\alpha$ and $\mathbf{x}_1$ is α in Figure 5. We also observe that the angle formed by $\mathbf{r}_\alpha$ and $\mathbf{x}_2$ is $(\alpha - \phi)$. Hence, we get

$$\frac{\mathbf{r}_\alpha}{r_\alpha} \cdot \mathbf{x}_2 = \cos(\alpha - \phi). \tag{17}$$

Replacing the term $\frac{\mathbf{r}_\alpha}{r_\alpha} \cdot \mathbf{x}_2$ in (16) by (17) gives

$$\dot{r}_\alpha = \dot{s}\cos(\alpha - \phi) - v_1\cos(\alpha). \tag{18}$$

We can now find $\dot{r}_\alpha$ and $\dot{s}$. Substituting the term $\dot{r}_\alpha$ in (13) for (18), we obtain

$$\dot{s}\cos(\phi) \;\; = \;\; v_1(1 + sin(\alpha)r_\alpha u) + \cos(\alpha)(\dot{s}\cos(\alpha - \phi) - v_1\cos(\alpha)). \tag{19}$$

Therefore, we obtain the time derivative of arc length $\dot{s}$ as

$$\dot{s} = \frac{v_1(r_\alpha u + \sin(\alpha))}{\sin(\alpha - \phi)}. \tag{20}$$

The term $\dot{s}$ in (18) can be replaced by $\dot{s}$ in (20) to get $\dot{r}_\alpha$ as follows:

$$\dot{r}_\alpha = v_1\frac{\sin(\phi) + r_\alpha u\cos(\alpha - \phi)}{\sin(\alpha - \phi)}. \tag{21}$$

Now, let us find the equation for $\dot{\phi}$. From Figure 5, we can see that the angle formed by $\mathbf{x}_1$ and $\mathbf{y}_2$ is $(\frac{\pi}{2} - \phi)$. This further implies that

$$\sin(\phi) = \mathbf{x}_1 \cdot \mathbf{y}_2. \tag{22}$$

Also, in Figure 5, the angle formed by $\mathbf{r}_\alpha$ and $\mathbf{y}_2$ is $(\frac{\pi}{2} + \alpha - \phi)$ so that

$$\mathbf{r}_\alpha \cdot \mathbf{y}_2 = -r_\alpha\sin(\alpha - \phi). \tag{23}$$

Differentiate (17) with respect to time on both sides to obtain

$$\sin(\alpha - \phi) \cdot \dot{\phi} = \frac{\dot{s} - v_1\cos(\phi)}{r_\alpha} - \frac{\dot{s}\cos(\alpha - \phi) - v_1\cos(\alpha)}{r_\alpha} \cdot \cos(\alpha - \phi) - \dot{s}\kappa\sin(\alpha - \phi), \tag{24}$$

where we have used (6), (9), (17), (18), and (23). Replacing $\dot{s}$ in (24) by (20), we derive the equation for $\dot{\phi}$ as

$$\dot{\phi} = v_1 \left(-\frac{\kappa \sin(\alpha)}{\sin(\alpha - \phi)} + u(1 - \frac{r_\alpha \kappa}{\sin(\alpha - \phi)}) \right). \tag{25}$$

For the Sting-I autonomous vehicle, the sensor on each side of the vehicle is installed such that $\alpha = \frac{\pi}{2}$. In this case, (20) is simplified to

$$\dot{s} = \frac{v_1(r_\alpha u + 1)}{\cos(\phi)}, \tag{26}$$

equation (21) is simplified to

$$\dot{r}_\alpha = v_1 \tan(\phi)(1 + r_\alpha u), \tag{27}$$

and (25) is simplified to

$$\dot{\phi} = v_1 \left(-\frac{\kappa}{\cos(\phi)} + u(1 - \frac{r_\alpha \kappa}{\cos(\phi)}) \right). \tag{28}$$

The system equations are significantly different from the equations for the closest point in [89].

2.2 Controller Design and Convergence Analysis

2.2.1 Lyapunov Function

Consider the Lyapunov function candidate:

$$V_1 = -\ln(\cos(\phi)) + h(r_\alpha), \tag{29}$$

where $h(r_\alpha)$ satisfies the following conditions:

1. $\frac{\partial h(r_\alpha)}{\partial r_\alpha} = f(r_\alpha)$, where $f(r_\alpha)$ is a Lipschitz continuous function on $(0,\infty)$, so that $h(r_\alpha)$ is continuously differentiable on $(0,\infty)$.

2. $\lim_{r_\alpha \to 0} f(r_\alpha) = -\infty$, which leads to $\lim_{r_\alpha \to 0} h(r_\alpha) = \infty$. This is needed to blow up V_1 as the moving vehicle approaches collision with the boundary curve.

3. $f(r_\alpha) = 0$ at the point where $r_\alpha = r_0$. At this point, $h(r_\alpha)$ has the local minimum, since r_0 is the desired relative distance between the vehicle and the boundary curve.

4. $\lim_{r_\alpha \to \infty} h(r_\alpha) = \infty$. Both this condition and the form of V_1 suggest that V_1 is radially unbounded (In other words, $V_1 \to \infty$ as $\|\phi\| \to \pi/2$, as $r_\alpha \to 0$, or as $r_\alpha \to \infty$).

Observe that V_1 given by (29) is continuously differentiable because of (7). The term $\ln(\cos(\phi))$ penalizes misalignment between the heading vector of the vehicle and the tangent vector to the boundary curve at the detected point. The term $h(r_\alpha)$ in (29) deals with the separation between the moving vehicle and the boundary curve. In short, V_1 is designed to make the vehicle converge to the relative position where $r_\alpha = r_0$ and $\phi = 0$. This form of Lyapunov function has also been used in curve tracking using the closest point information in [89] and [88].

For the point detected by the fixed center ray at the angle $\alpha = \pi/2$, our candidate $f(r_\alpha)$ is

$$f(r_\alpha) = \frac{-1}{r_\alpha} + \frac{1}{r_0}. \tag{30}$$

Furthermore, the corresponding $h(r_\alpha)$ is

$$h(r_\alpha) = -\ln(r_\alpha) + \frac{r_\alpha}{r_0} + \ln(r_0) - 1, \tag{31}$$

which satisfies the conditions for $h(r_\alpha)$.

The time derivative of V_1 is now

$$\dot{V}_1 = v_1 \tan(\phi)[u(1 - \frac{r_\alpha \kappa}{\cos(\phi)} + f(r_\alpha) r_\alpha) - \frac{\kappa}{\cos(\phi)} + f(r_\alpha)], \tag{32}$$

where we have used (27), (28), and (29). We now assume that the speed $v_1 > 0$ is a constant and design the steering control u so that $\dot{V}_1 \leq 0$.

2.2.2 Tracking Control for Convex Curves

We first consider the case when the curve is convex and curving away from the vehicle. In this case, we have $\kappa < 0$.

One choice of u which leads to $\dot{V}_1 \leq 0$ is

$$u_1 = \frac{v_1\kappa - \cos(\phi)(v_1 f(r_\alpha) + \mu \sin(\phi))}{v_1(\cos(\phi) + f(r_\alpha) r_\alpha \cos(\phi) - r_\alpha \kappa)}, \tag{33}$$

where $\mu > 0$ is a constant. The time derivative of V_1 in (32) with u given by (33) is

$$\dot{V}_1 = -\mu \frac{\sin^2(\phi)}{\cos(\phi)} \leq 0 \tag{34}$$

where (7) is used. Thus, $\dot{V}_1 \leq 0$ and $\dot{V}_1 = 0$ if and only if $\sin(\phi) = 0$. By (7), we see that $\dot{V}_1 = 0$ if and only if $\phi = 0$.

From now on, we refer to the case where the denominator of a control law is zero as the *singularity* of the control law. It *seems possible* that the control law given by (33) is singular when $\cos(\phi) = \frac{r_\alpha \kappa}{1 + f(r_\alpha) r_\alpha}$. Using (30), the singularity of (33) occurs when

$$\cos(\phi) = \frac{r_\alpha \kappa}{1 + f(r_\alpha) r_\alpha} = r_0 \kappa. \tag{35}$$

Therefore, in the case where κ, the curvature of the lane at the detected point, is equal to or smaller than zero in (35), the denominator of (33) will never be zero since $\cos(\phi) > 0$.

Theorem 1. *Consider the case where the boundary curve is convex, i.e., $\kappa < 0$. Then, using the steering control law in (33), the vehicle satisfying (7) with constant speed v_1 converges to the state where it tracks the curve at the distance r_0 without collision.*

Proof. For each trajectory that initially satisfies (7) and $r_\alpha > 0$, there exists a compact sublevel set Ω of V_1 such that the trajectory remains in Ω for all future time. Then by LaSalle's Invariance Principle [47], the trajectory converges to the largest invariant

set M within the set E that contains all points in Ω where $\dot{V}_1 = 0$. The set E in this case is the set of all points in Ω such that $\phi = 0$. Note that $\phi = 0$ implies $\dot{r}_\alpha = 0$ using (27). Thus, at any point in E, the dynamics are expressed as

$$\dot{r}_\alpha = 0. \tag{36}$$

Since the trajectory converges to the maximum invariant set M within the set E where $\phi = 0$, we have $\dot{\phi} \to 0$. Therefore, replacing the term $\dot{\phi}$ in (28) by 0 gives

$$v_1 u_1 - v_1(r_\alpha u_1 + 1)\kappa = 0. \tag{37}$$

Hence, the control input u_1 in the invariant set M is

$$u_1 = \frac{\kappa}{1 - r_\alpha \kappa}. \tag{38}$$

When we substitute ϕ in (33) for 0, the corresponding control input is

$$u_1 = \frac{\kappa - f(r_\alpha)}{1 + f(r_\alpha) r_\alpha - r_\alpha \kappa}. \tag{39}$$

u_1 in (39) should be equal to u_1 in (38), because both u_1 are control inputs in the invariant set M. Thus, we obtain

$$\frac{\kappa}{1 - r_\alpha \kappa} = \frac{\kappa - f(r_\alpha)}{1 + f(r_\alpha) r_\alpha - r_\alpha \kappa}, \tag{40}$$

which implies

$$(\kappa - f(r_\alpha))(1 - r_\alpha \kappa) = \kappa + f(r_\alpha) r_\alpha \kappa - r_\alpha \kappa^2. \tag{41}$$

Therefore, $f(r_\alpha)$ must satisfy

$$f(r_\alpha) = 0. \tag{42}$$

The moving vehicle converges to the position at a distance from the boundary curve given by the zero of the function $f(\cdot)$. Therefore, the largest invariant set contained in E is expressed as

$$M = \{(r_\alpha, \phi) | \phi = 0, f(r_\alpha) = 0\}. \tag{43}$$

Thus, we can conclude that (r_α, ϕ) converges to the equilibrium where $r_\alpha = r_0$ and $\phi = 0$. □

2.2.3 Control Laws for Concave Curve with Bounded Curvature

We consider the case when the curve is concave, i.e., curving towards the vehicle. In this case, we have $\kappa > 0$. It is possible that the control law given by (33) is singular when the denominator of u_1 equals to zero, i.e., $\cos(\phi) = \frac{r_\alpha \kappa}{1+f(r_\alpha)r_\alpha}$. However, (35) shows that if the curvature κ is bigger than $\frac{1}{r_0}$, then no singularity happens because $|\cos\phi| \leq 1$.

In the real experimental environments, it is necessary for the vehicle to follow a concave curve whose curvature is small. We argue that in this case, the singularity exists regardless of the choice of $f(r_\alpha)$. Figure 6 shows possible graphs of $1 + f(r_\alpha)r_\alpha$ and $r_\alpha\kappa$. When (30) is used as $f(r_\alpha)$, we get $1+f(r_\alpha)r_\alpha = \frac{r_\alpha}{r_0}$. Therefore, the straight line connecting the origin and (r_0,1) represents $1+f(r_\alpha)r_\alpha$ when (30) is used as $f(r_\alpha)$. In Figure 6, regardless of $f(r_\alpha)$, $1 + f(r_\alpha)r_\alpha$ is a continuous function which is equal to 1 when $r_\alpha = r_0$. Also, regardless of the decreasing rate of $f(r_\alpha)$ as $r_\alpha \to 0$, we can assure that $\lim_{r_\alpha \to 0} 1 + f(r_\alpha)r_\alpha \leq 1$. As $r_\alpha \downarrow r_0$, we see that $f(r_\alpha)$ and r_α both decrease to make $(1 + f(r_\alpha)r_\alpha)$ decrease for any choice of $f(r_\alpha)$. Meanwhile, possible $r_\alpha\kappa$ are plotted with straight lines. If the curvature κ is less than $\frac{1}{r_0}$, then these straight lines will be below the curve that represents $(1+f(r_\alpha)r_\alpha)$, regardless of what $f(r_\alpha)$ is. Therefore, $\frac{r_\alpha\kappa}{1+f(r_\alpha)r_\alpha} < 1$ and $\cos\phi = \frac{r_\alpha\kappa}{1+f(r_\alpha)r_\alpha}$ always has a solution for ϕ. This singularity cannot be removed by changing $f(r_\alpha)$.

2.2.4 The Safety Zone

Due to (35), if $|\phi| < \arccos(r_0\kappa_M)$, where κ_M is the upper bound of κ, then $\cos\phi > r_0\kappa$, which implies that the singularity will never happen. Thus, we define the set

$$U = \{(r_\alpha, \phi) | V_1(r_\alpha, \phi) < -\ln(|r_0\kappa_M|)\} \tag{44}$$

as the *safety zone*. Note that we assume $\kappa_M r_0 < 1$ since otherwise the desired distance is too far away from the curve, which makes tracking meaningless. The controller (33) is used inside the safety zone. Since this controller yields $\dot{V}_1 \leq 0$, we conclude that

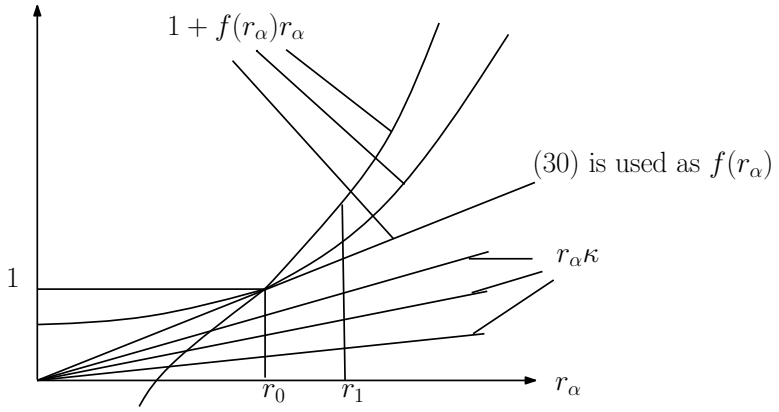


Figure 6: The graphs for $1 + f(r_\alpha)r_\alpha$ and $r_\alpha\kappa$. The control law given by (33) is singular when $\cos(\phi) = \frac{r_\alpha\kappa}{1+f(r_\alpha)r_\alpha}$. We argue that this singularity cannot be removed by choosing $f(r_\alpha)$ if the curvature κ is less than $\frac{1}{r_0}$.

once the vehicle under control enters the safety zone U, it will never leave. Therefore, according to Theorem 1, the curve-tracking behavior is stabilized without collision if the vehicle starts *inside* the safety zone.

2.2.5 Switching Control that Aims for the Safety Zone

When the vehicle is initially out of the safety zone, the vehicle may come close to the set where $\cos(\phi) = r_0\kappa$ during its movements. This indicates that the control law in (33) cannot be applied due to the singularity.

The positions of the vehicle when singularities occur are plotted as rectangles in Figure 7. The singularities occur when the vehicle is positioned on the line l, and the angle ϕ satisfies $\cos(\phi) = r_0\kappa$. The angle $\phi < 0$ is measured counterclockwise from $\mathbf{x}_2$, tangent vector to the boundary curve at the detected point, to $\mathbf{x}_1$, the heading direction of the vehicle. When $\phi > 0$, the vehicle will be at the same position but heading away from the boundary curve. In Figure 7, r_k denotes the radius of the osculating circle at the detected point p so that $r_k = 1/\kappa$. The vehicle's desired curve is plotted as d that has r_0 distance from the boundary curve. In the illustrated case, the controller design problem should be reconsidered because the goal of the controller now is to steer into the safety zone. Intuitively, this means that the vehicle should be

steered away from the boundary curve promptly, which is a natural behavior when we drive our cars on a collision course to a concave wall. Therefore, we now design controllers so that the vehicle enters the safety zone U in finite time.

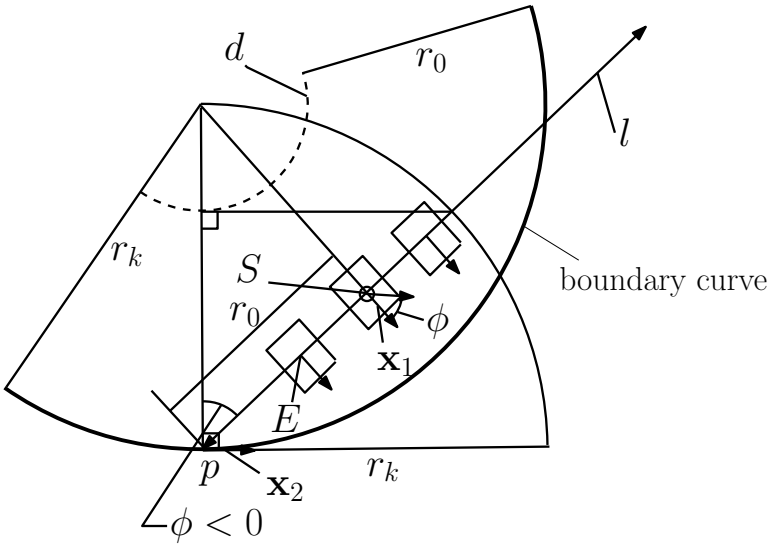


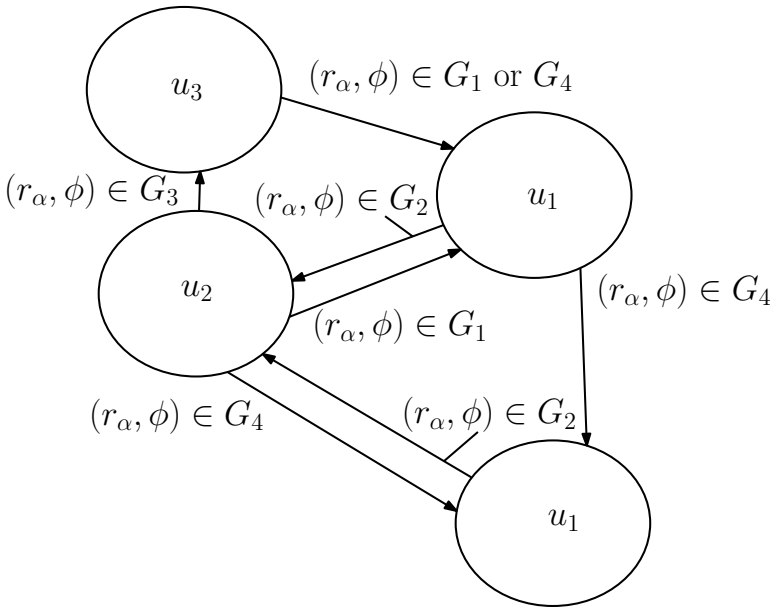
Figure 7: The positions of the vehicle when singularities occur. The positions of the vehicle when singularities occur are depicted as rectangles. The singularities occur when the vehicle is positioned on the line l, and the angle ϕ satisfies $\cos(\phi) = r_0\kappa$.

We develop a switching system as depicted in Figure 8 to steer the system into the safety zone in finite time. Four cases are distinguished, which correspond to four sets (G_1, G_2, G_3, and G_4) defined as follows:

$$\begin{aligned} G_1 &= \{(r_\alpha, \phi) | \| \cos(\phi) - r_0\kappa \| > \epsilon \text{ but } (r_\alpha, \phi) \notin U\} \\ G_2 &= \{(r_\alpha, \phi) | \epsilon_2 < \| \cos(\phi) - r_0\kappa \| \leq \epsilon\} \\ G_3 &= \{(r_\alpha, \phi) | \| \cos(\phi) - r_0\kappa \| \leq \epsilon_2\} \\ G_4 &= U, \end{aligned} \tag{45}$$

where $\epsilon_2 < \epsilon$.

Three control laws are designed for these four cases. When the system states are in G_1 or G_4, we use u_1 in (33). When the states enter G_2 from G_1, we switch to u_2



u_1	$u_1 = \frac{v_1\kappa - \cos(\phi)(v_1 f(r_\alpha) + \mu \sin(\phi))}{v_1(\cos(\phi) + f(r_\alpha) r_\alpha \cos(\phi) - r_\alpha \kappa)}$.
u_2	$u_2 = \frac{v_1\kappa - \cos(\phi)(v_1 f(r_\alpha) + \mu_2 \sin(\phi))}{v_1(\cos(\phi) + f(r_\alpha) r_\alpha \cos(\phi) - r_\alpha \kappa)}$.
u_3	$u_3 = \frac{-\mu_3 \sin(\phi) + \kappa v_1 r_\alpha}{v_1 r_\alpha(\cos(\phi) - r_\alpha \kappa)}$.

Figure 8: The switching control strategy used to enter the safety zone. u_1 in (33) is used in normal situations, i.e., when the states are in G_1 or G_4. We switch to u_2 in (46) when the states enter G_2 and switch to u_3 in (48) when the states enter G_3.

which is

$$u_2 = \frac{v_1\kappa - \cos(\phi)(v_1 f(r_\alpha) + \mu_2 \sin(\phi))}{v_1(\cos(\phi) + f(r_\alpha) r_\alpha \cos(\phi) - r_\alpha \kappa)}, \tag{46}$$

where the only difference between u_1 and u_2 is that μ_2 in u_2 is much bigger than μ in u_1. The time derivative of V_1 under control u given by (46) is

$$\dot{V}_1 = -\mu_2 \frac{\sin^2(\phi)}{\cos(\phi)} \leq 0. \tag{47}$$

When the states of the system enter G_3 from G_2, we switch to u_3:

$$u_3 = \frac{-\mu_3 \sin(\phi) + \kappa v_1 r_\alpha}{v_1 r_\alpha(\cos(\phi) - r_\alpha \kappa)}, \tag{48}$$

where $\mu_3 > 0$ is a constant. Under this controller, we have

$$\dot{\phi} = -\frac{\mu_3 \tan(\phi)}{r_\alpha}, \tag{49}$$

where (28) is used. Using (49), we have $\phi \to 0$ as $t \to \infty$. This implies that the system states will get out of G_3 and then out of G_2 in finite time. We switch back to u_1 after the states enter either G_1 or G_4. Note that by Theorem 1, once the states enter G_4, they will stay in G_4 and converge to the desired values.

We now prove convergence of the system under the switching control laws illustrated in Figure 8. The idea is that the Lyapunov function V_1 may increase under u_3, but such increase will be compensated by the decrease of V_1 under u_2. Hence, the overall effect is that the Lyapunov function decreases until the system reaches G_4. Some notations and technical conditions are needed to rigorously state and prove the results.

It is uninteresting if the states never enter G_3, since V_1 will decrease until G_4 is reached. Therefore, we discuss the most general case, i.e., the states of the system enter G_3 for a number of times. In order to enter G_3, the system must enter G_2 first. We use the notations t_1^i to indicate the time when the system enters G_2, t_2^i to indicate the time when the system enters G_3, and t_3^i to indicate the time when the system

enters either G_1 or G_4. The index i is used to distinguish among multiple entries. If the states enter G_2, then t_1^i, t_2^i, and t_3^i happen in sequence.

The following technical assumptions are needed.

(T1) The curvature κ is bounded above by $\kappa_M > 0$.

(T2) The desired distance r_0 satisfies that $r_0\kappa_M < 1$.

(T3) Define $\zeta = v_1\| - \arccos(\kappa_M r_0 + \epsilon) + \arccos(\kappa_M r_0 - \epsilon_2)\| + \epsilon_3$, where $\epsilon_3 > 0$ is a constant. The gains μ_2 in u_2 and μ_3 in u_3 satisfy $\mu_2\mu_3(t_2^i - t_1^i) > \frac{\zeta r_0\kappa_M}{1-(r_0\kappa_M)^2}$ for all i.

Assumptions (T1) and (T2) put mild constraints on the curve to follow. Assumption (T3) is the key technical assumption. This assumption is satisfied if μ_2 or μ_3 is sufficiently large.

Theorem 2. *Consider the system defined by (27) and (28) governing the relative distance and heading angle between the vehicle and the detected point. Suppose the vehicle travels at constant speed v_1. Under the switching strategy in Figure 8, with assumptions (T1)-(T3) satisfied, the states of the vehicle enter G_4 in finite time.*

Proof. We organize our proofs in two steps:

1. show that when u_3 is used, V_1 will increase a finite amount bounded above.
2. show that when u_2 is used, V_1 will decrease more than the upper bound for its increase under u_3.

1. Estimate the upper bound for the increase of V_1 under u_3.

The time derivative of V_1 under u_3 is

$$\dot{V}_1 = -\tan(\phi)[u_3(\frac{v_1 r_\alpha \kappa}{\cos(\phi)} - v_1\frac{r_\alpha}{r_0}) + \frac{v_1}{\cos(\phi)}\kappa - v_1(\frac{-1}{r_\alpha} + \frac{1}{r_0})], \tag{50}$$

where (30) is used as $f(r_\alpha)$. Notice that u_3 is used only in the small neighborhood of $\cos(\phi) = \kappa r_0$. Replacing $\cos(\phi)$ in (50) by κr_0, we get

$$\dot{V}_1 = -\tan(\phi)\frac{v_1}{r_\alpha}. \tag{51}$$

If $\sin(\phi) \geq 0$, then $\dot{V}_1 \leq 0$ is guaranteed. This implies that V_1 decreases while u_3 is used. This case is uninteresting.

The case that V_1 may increase is shown in Figure 7. We now estimate the increase of V_1 while u_3 is used as the control law.

$$V_1(t_3^i) - V_1(t_2^i) = -v_1 \int_{t_2^i}^{t_3^i} \frac{\tan(\phi(t))}{r_\alpha(t)} dt. \tag{52}$$

We can change (52) to integration with respect to ϕ as

$$V_1(t_3^i) - V_1(t_2^i) = \frac{v_1}{\mu_3} \int_{\phi(t_2^i)}^{\phi(t_3^i)} d\phi = \frac{v_1}{\mu_3}(\phi(t_3^i) - \phi(t_2^i)), \tag{53}$$

where (49) is used. The controller u_3 is applied from the instant when $|\cos(\phi) - \kappa r_0| = \epsilon_2$ to the instant when $|\cos(\phi) - \kappa r_0| = \epsilon$, where $\epsilon_2 < \epsilon$. Therefore, we get $|\cos(\phi(t_2^i)) - \kappa r_0| = \epsilon_2$ and $|\cos(\phi(t_3^i)) - \kappa r_0| = \epsilon$. Thus, when $\phi < 0$, possible values of ϕ can be listed as follows:

$$\begin{aligned} \phi(t_2^i) &= -\arccos(\kappa r_0 \pm \epsilon_2) < 0 \\ \phi(t_3^i) &= -\arccos(\kappa r_0 \pm \epsilon) < 0. \end{aligned} \tag{54}$$

We plot these possible values on Figure 9. Within the interval of $-\pi/2 < \phi < 0$, $\cos(\phi)$ increases as ϕ increases. Thus, we get $-\arccos(\kappa r_0 + \epsilon) > -\arccos(\kappa r_0 - \epsilon)$, and $-\arccos(\kappa r_0 + \epsilon_2) > -\arccos(\kappa r_0 - \epsilon_2)$. Therefore, we conclude that

$$\phi(t_3^i) - \phi(t_2^i) \leq \max(\phi(t_3^i)) - \min(\phi(t_2^i)) = -\arccos(\kappa r_0 + \epsilon) + \arccos(\kappa r_0 - \epsilon_2). \tag{55}$$

Figure 9 compares between

$$-\arccos(\kappa r_0 + \epsilon) + \arccos(\kappa r_0 - \epsilon_2)$$

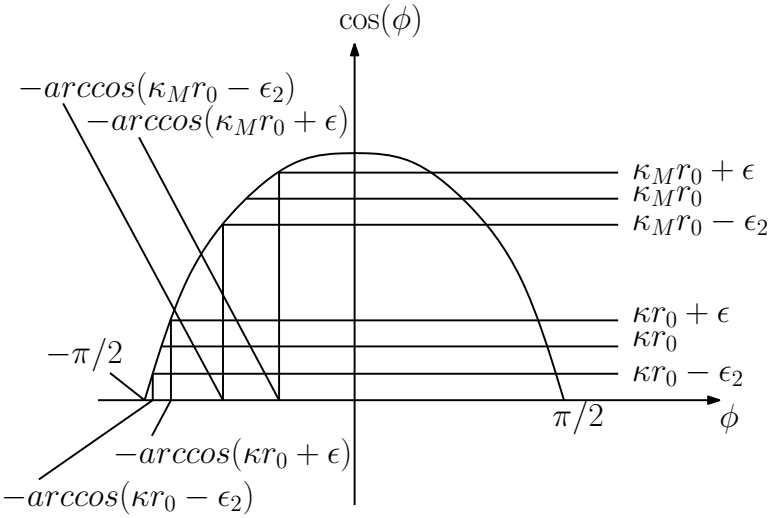


Figure 9: The graph of $\cos(\phi)$ with respect to ϕ. The slope of $\cos(\phi)$ with respect to ϕ, $\frac{d\cos(\phi)}{d\phi} = -\sin(\phi)$, monotonously decreases to zero as ϕ goes to zero in the interval of $-\pi/2 < \phi < 0$. Therefore, as seen in this figure, we get $-\arccos(\kappa r_0 + \epsilon) + \arccos(\kappa r_0 - \epsilon_2) \leq -\arccos(\kappa_M r_0 + \epsilon) + \arccos(\kappa_M r_0 - \epsilon_2)$.

and

$$-\arccos(\kappa_M r_0 + \epsilon) + \arccos(\kappa_M r_0 - \epsilon_2).$$

The slope of $\cos(\phi)$ with respect to ϕ is $\frac{d\cos(\phi)}{d\phi} = -\sin(\phi)$. It monotonously decreases to zero as ϕ goes to zero in the interval of $-\pi/2 < \phi < 0$. Thus, as seen in Figure 9, we get

$$-\arccos(\kappa r_0 + \epsilon) + \arccos(\kappa r_0 - \epsilon_2) \leq -\arccos(\kappa_M r_0 + \epsilon) + \arccos(\kappa_M r_0 - \epsilon_2).$$

According to (55), we further deduce that

$$\phi(t_3^i) - \phi(t_2^i) \leq -\arccos(\kappa_M r_0 + \epsilon) + \arccos(\kappa_M r_0 - \epsilon_2). \tag{56}$$

Now, using (53) and (56), we can derive the upper bound for the increase of V_1 while u_3 is used.

$$V_1(t_3^i) - V_1(t_2^i) \leq \frac{v_1}{\mu_3}(-\arccos(\kappa_M r_0 + \epsilon) + \arccos(\kappa_M r_0 - \epsilon_2)) < \frac{\zeta}{\mu_3}, \tag{57}$$

where ζ is defined in assumption (T3).

2. We show that the decrease of V_1 under u_2 is larger than the upper bound for the increase of V_1 under u_3.

We compute the required length of the time interval when u_2 is used so that V_1 decreases more than ζ/μ_3. In other words,

$$V_1(t_2^i) - V_1(t_1^i) = \int_{t_1^i}^{t_2^i} \dot{V}_1 dt < -\frac{\zeta}{\mu_3}, \tag{58}$$

where t_1^i and t_2^i represent the beginning and the end of the interval when u_2 is used respectively. Hence, using (47), we require that

$$\int_{t_1^i}^{t_2^i} \frac{\mu_2 \sin^2(\phi)}{\cos(\phi)} dt = (t_2^i - t_1^i) \frac{\mu_2 \sin^2(\phi(\tau))}{\cos(\phi(\tau))} > \frac{\zeta}{\mu_3}, \tag{59}$$

where $\tau \in [t_1^i, t_2^i]$ and the Mean Value Theorem is applied. Furthermore, the required length of the time interval under u_2 in order to decrease V_1 more than ζ/μ_3 is

$$(t_2^i - t_1^i) > \frac{\zeta \cos(\phi(\tau))}{\mu_2 \mu_3 \sin^2(\phi(\tau))}. \tag{60}$$

As seen in Figure 8, u_2 is used in the near-singular state. Thus, we can see that $\cos(\phi(\tau)) \approx r_0\kappa \leq r_0\kappa_M < 1$ using assumption (T2). Therefore, we get

$$\frac{\sin^2(\phi(\tau))}{\cos(\phi(\tau))} = \frac{1}{\cos(\phi(\tau))} - \cos(\phi(\tau)) \geq \frac{1}{r_0\kappa_M} - r_0\kappa_M > 0. \tag{61}$$

This is equivalent to

$$\frac{\cos(\phi(\tau))}{\sin^2(\phi(\tau))} \leq \frac{r_0\kappa_M}{(1 - (r_0\kappa_M)^2)}. \tag{62}$$

Multiplying both sides of (62) by $\frac{\zeta}{\mu_2\mu_3}$, we derive

$$\frac{\zeta \cos(\phi(\tau))}{\mu_2\mu_3 \sin^2(\phi(\tau))} \leq \frac{\zeta r_0\kappa_M}{\mu_2\mu_3(1 - (r_0\kappa_M)^2)}. \tag{63}$$

Therefore, using (60) and (63), if

$$(t_2^i - t_1^i) > \frac{\zeta r_0\kappa_M}{\mu_2\mu_3(1 - (r_0\kappa_M)^2)}, \tag{64}$$

we can guarantee that the decrease of V_1 under u_2 is bigger than the increase of V_1 under u_3 by an amount of ϵ_3/μ_3. We can then conclude that switching among u_1, u_2, and u_3 will make the system enter the safety zone in finite time. □

In Figure 10, a typical switching process is plotted. u_1 is used from 0 to t_1^i, u_2 is used from t_1^i to t_2^i, u_3 is used from t_2^i to t_3^i, and u_1 is used again after t_3^i. In assumption (T3), we use arbitrarily large μ_2 or μ_3 so that the interval of using u_2 is long enough to overcome the increase of V_1 under u_3. Therefore, V_1 always decreases more than it increases.

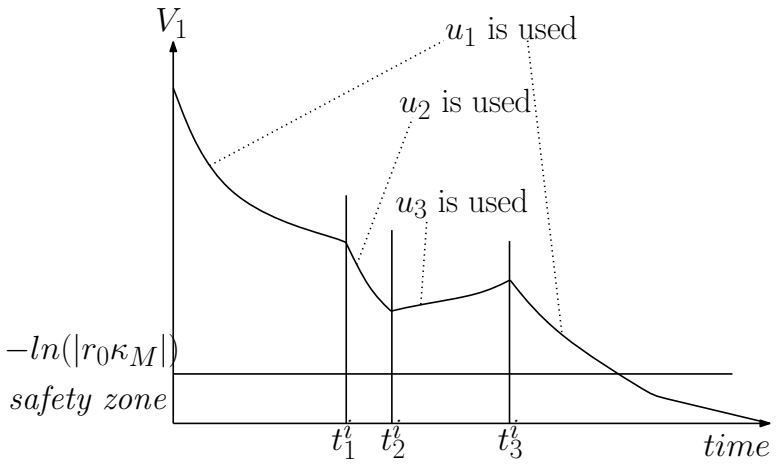


Figure 10: The Lyapunov function V_1 in a typical case of switching control. u_1 in (33) is used from 0 to t_1^i, u_2 in (46) is used from t_1^i to t_2^i, u_3 in (48) is used from t_2^i to t_3^i, and u_1 is used from t_3^i to final time.

In the case where $r_\alpha = r_0$ and $\cos(\phi) = r_0\kappa$, we have the singularity of u_1, u_2, and u_3 at the same time. This is the *common singularity* that occurs when the vehicle is at the point S in Figure 7. As seen in Figure 7, $\mathbf{x}_1$, the heading direction of the vehicle, at S is normal to the desired curve d. This singularity will not happen if the vehicle is in the safety zone. Since it happens only at the point S and the vehicle has constant speed, we conclude that the vehicle will not likely be in this state unless it starts initially in this state. The authors of [89] also mentioned that the moving vehicle should not be initially heading directly toward the boundary curve when control laws based on closest point information are applied.

2.3 Simulation Results

We implement our feedback control laws in MATLAB, as well as in the three dimensional simulation program used in the Georgia Tech urban grand challenge system. Our three dimensional simulation program is based on Player, Stage, and Gazebo that are three pieces of software developed for robotic simulation projects.

2.3.1 MATLAB Simulation Results

Figure 11 shows a vehicle following a closed boundary curve in a clockwise direction. We vary the vehicle's initial x-coordinate from -8 to 8, and y-coordinate from -6 to 6, with initial orientation $3\pi/4$ measured counterclockwise from the x-axis. The desired separation between the vehicle and the boundary curve is set to 0.5 distance units, and v_1, the velocity of the vehicle, is 0.5 distance units per unit time.

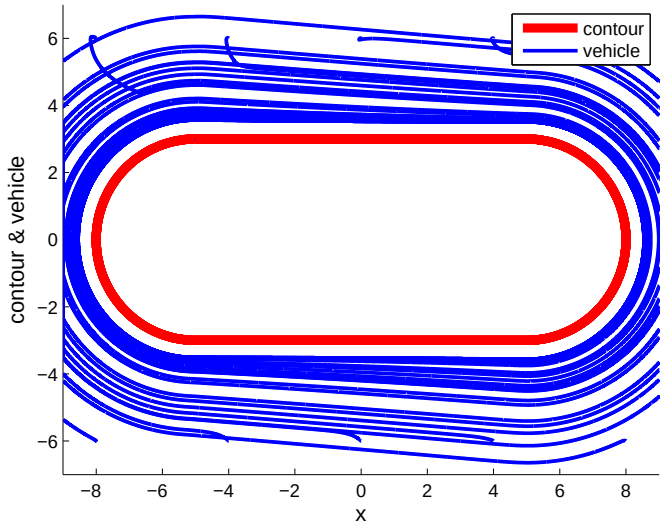


Figure 11: A vehicle following a closed boundary curve in a clockwise direction. We vary the vehicle's initial x-coordinate from -8 to 8, and y-coordinate from -6 to 6 with initial orientation $3\pi/4$ measured counterclockwise from the x-axis.

Figure 12 is a MATLAB simulation showing the result of using switching controllers to overcome the singularity. In order to compare with Figure 7, the vehicle moves toward a concave curve initially and the curvature of which is 1. Also, the

desired curve has 0.5 distance units from the obstacle. The initial position and the heading direction of the vehicle are the same as those for the vehicle at E in Figure 7. We can find that using switching controllers, the autonomous vehicle converges to the desired curve as expected.

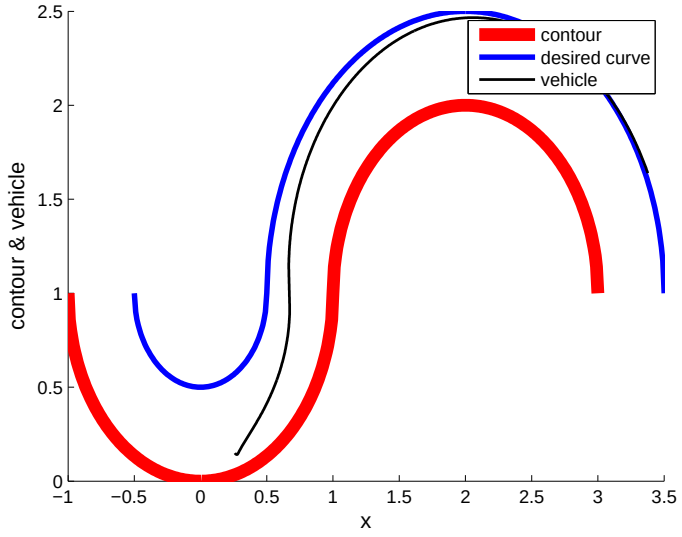


Figure 12: The result of using switching controllers to overcome the singularity. The initial position and the heading direction of the vehicle are the same as those for the vehicle at E in Figure 7. Switching occurs when the vehicle is at the near-singular state, and the vehicle is steered away from the boundary curve promptly.

2.3.2 Verification Using the Three Dimensional Simulation Program

Our switching controllers are verified using the three dimensional simulation program developed for the Georgia Tech urban grand challenge system. To estimate the curvature at the detected point using a group of rays around the center ray, we use the following estimation method.

Let P_n represent the detected point, which is the point on the obstacle boundary detected by a center ray. Suppose that there are two rays around the center ray with the window size w. Let P_{n-w} and P_{n+w} denote two points on the boundary curve detected using the two rays. The method for curvature estimation was proposed in [12] as follows. Let $a = \|P_n - P_{n-w}\|, b = \|P_{n+w} - P_n\|, c = \|P_{n+w} - P_{n-w}\|$, and

$s = (a + b + c)/2$. We draw the unique circle passing all three points. By applying Heron's formula, the curvature of such a circle is

$$\|\hat{\kappa}(s)\| = \frac{4\sqrt{s(s-a)(s-b)(s-c)}}{abc}. \tag{65}$$

In [12], it was proved that $\hat{\kappa}$ is a good estimate of κ when the difference $(a - b)$ is sufficiently small. We refer to this estimate as the *geometric estimate* of curvature. In [93], the authors derived the extended version of this geometric curvature estimate. For example,

$$\bar{\kappa}(n) = \frac{1}{3}\sum_{w=7}^{9} \hat{\kappa}(P_{n-w}, P_n, P_{n+w}), \tag{66}$$

where $\hat{\kappa}(P_{n-w}, P_n, P_{n+w})$ denotes the geometric estimate of curvature obtained at P_n with the window size w. In [93], it was shown that using a larger window size eliminates the need for Gaussian filtering. In our simulation experiments, (66) is taken as a method to estimate the curvature of the lane at the detected point P_n.

Figure 13 to Figure 17 show the simulation results using this three dimensional simulation program. The desired distance r_0 is set to 10 distance units, and the vehicle's velocity v_1 is set to 6 distance units per second.

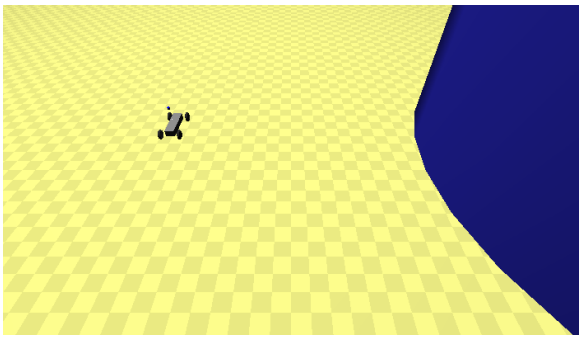


Figure 13: The initial position of the vehicle in the three dimensional simulation program. On the right side of the vehicle, we can find a cylinder shaped obstacle. The diameter and the height of the obstacle are set to 40 distance units and 20 distance units respectively.

Figure 13 shows the simulation environment where a cylinder shaped obstacle is built on the right side of the initial position of the vehicle. The diameter and the

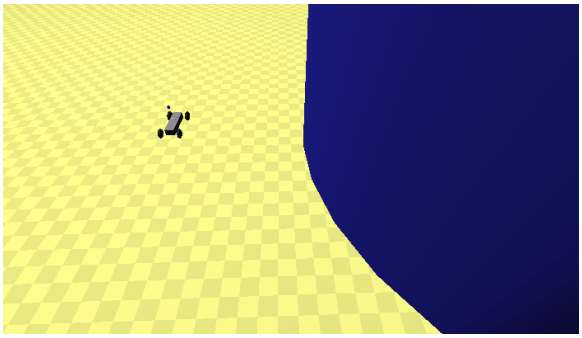


Figure 14: The final position of the vehicle in the three dimensional simulation program. The vehicle converges to the position where r_α, the relative distance from the obstacle, is almost 10 distance units as we desired.

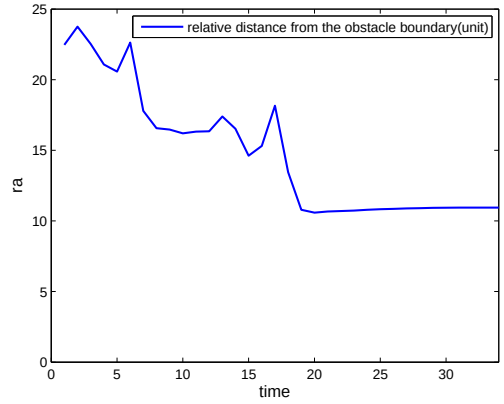


Figure 15: r_α, the vehicle's relative distance from the obstacle, with respect to time.

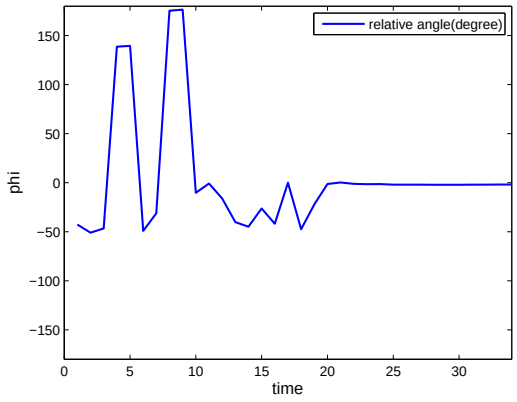


Figure 16: The vehicle's relative heading angle, ϕ, with respect to time. ϕ converges to 0 as time goes on.

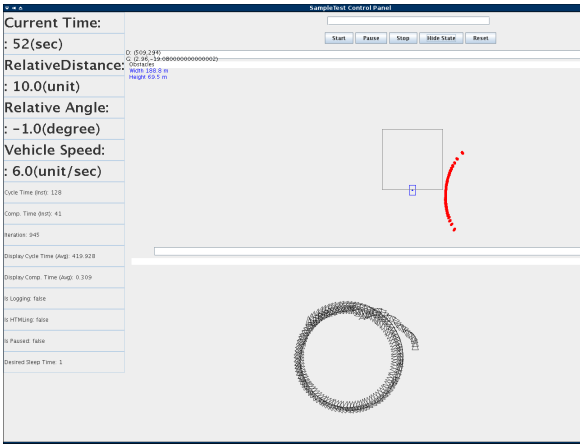


Figure 17: The control panel of the simulation at the final time. In this simulation, desired distance (r_0) is set to 10 distance units, and the vehicle's velocity (v_1) is set to 6 distance units per second. Accordingly, relative distance (r_α=10.0 distance units), relative angle (ϕ=-1.0 degree), and vehicle speed (v_1=6.0 distance units per second) are displayed on the left side of this control panel. On the right side of the panel, the planar trajectory of the vehicle is displayed as a circle, since we have a cylinder shaped obstacle.

height of the obstacle are set to 40 distance units and 20 distance units respectively. Figure 14 presents that the vehicle converges to the position where r_α, the relative distance from the obstacle, is almost 10 distance units as we desired. Figure 15 shows that r_α converges to r_0. Figure 16 shows that the vehicle's relative heading angle ϕ converges to 0 as time goes on. In Figure 16, the overshoot of ϕ is large initially, since switching control laws are used to overcome the singularity caused by the error in the curvature estimate using (66). Figure 17 displays the values on the control panel at the final time. On the right side of the panel, the trajectory of the vehicle projected to the plane is displayed.

2.4 Conclusions

In this chapter, we design curve-tracking control that uses information from rigidly mounted, narrow aperture range sensors. The key idea is to control the relative motion between the vehicle and the detected point, and to switch controllers to avoid singularities.

CHAPTER III

A PROVABLY COMPLETE EXPLORATION STRATEGY BY CONSTRUCTING VORONOI DIAGRAMS

This chapter addresses the problem of exploring an unknown workspace using one vehicle equipped with range sensors. Such a sensor has the ability to determine a point on obstacle boundary that is closest to the vehicle. We call such a point the *closest point*. If boundary curves appear on both the left and the right hand sides of the vehicle, then a closest point can be determined on each boundary. The path that has equal distances from these closest points is the Voronoi edge. All Voronoi edges form the Voronoi diagram that reveals the topological structure of the workspace. If the vehicle visits all Voronoi edges in the workspace, then we consider the workspace as being completely explored.

Inspired by the curve-tracing control in Chapter 2, we develop a provably convergent Lyapunov-based control law to track a Voronoi edge. Our Voronoi edge tracking control law is based on the shape dynamics derived in [89]. In [89] and [93], a gyroscopic feedback control law was developed to control the interaction between the vehicle and the closest point so that the vehicle follows the obstacle boundary either to the left or to the right. The closest point was also used for path following in [80]. Our curve tracking control law extends previous works by using information from the closest points on both sides of the vehicle. This results in a tracking behavior of a Voronoi edge between two obstacles.

Utilizing the Voronoi edge tracking behavior, we develop provably complete exploration algorithms, denoted as the *Boundary Expansion (BE) algorithms*, which enable the construction of a topological map based on Voronoi diagrams. Although many

results [82, 76, 15, 18, 14, 19, 73] exist in the literature regarding the construction of Voronoi diagrams, to our knowledge, the BE algorithms are unique, with provable completeness over a connected and compact workspace.

The BE algorithms are composed of two algorithms, denoted by algorithm 1 and algorithm 2 in this chapter. Applying algorithm 1, the trajectory of a vehicle constructs a simple closed curve that encloses an obstacle to its right. Then, using algorithm 2, the vehicle iteratively expands the explored area in a way that one obstacle is added to the area at a time. In this way, the vehicle constructs the Voronoi diagram by "expanding" the explored area in discrete and finite steps.

In the BE algorithms, only the graph structure representing the boundary of the explored area is maintained and updated based on two simple rules. There is no need to explicitly search for the shortest path in the graph as in many other exploration algorithms [18, 82, 76]. Hence, the BE algorithms may have lower computational load.

We implement the algorithms and the tracking control law using a miniature robot localizing itself based on an odometry system. The robot uses only Infrared(IR) sensors for range measurements. Simulations and experimental results demonstrate the effectiveness of both the tracking control law and the algorithms.

The chapter is organized as follows. Section 3.1 presents the workspace to be explored. Section 3.2 introduces the provably convergent control law to track Voronoi edges. Section 3.3 discusses the BE algorithms, and Section 3.4 provides proofs for the convergence of the BE algorithms. Section 3.5 analyzes the efficiency of the BE algorithms. Section 3.6 provides simulations and experimental results to demonstrate the effectiveness of both the control law and the exploration algorithms. Lastly, Section 3.7 provides conclusions.

3.1 The Workspace and Its Voronoi Diagram

Consider a connected and compact workspace $W \subset \mathcal{R}^2$ whose boundary, ∂W, is a regular curve. Let O_1,O_2,...O_{M-1} be $M-1$ disjoint and compact obstacles such that $O_i \subset W$. O_M is a "virtual" obstacle that bounds the workspace, i.e., $\partial W \subset \partial O_M$. We denote the set of obstacles S_O by $S_O = \{O_1, O_2, ..., O_M\}$.

Obeying the conventions established in the literature on Voronoi diagrams [2, 73, 61, 58, 14], we define the *Voronoi cell* for an obstacle O_i as the set of points that is closer to O_i than to any other obstacle in S_O for $i = 1, 2, ..., M$ i.e.

$$V(O_i) = \{q \in W | \min_{z \in O_i} \|z - q\| < \min_{z' \in O_i'} \|z' - q\|, \forall O_i' \in S_O \setminus O_i\}, \tag{67}$$

where $\|\cdot\|$ is the Euclidean norm in $\mathcal{R}^2$. $\partial V(O_i)$ is the boundary of the Voronoi cell for O_i, i.e., $V(O_i)$. Also, $\overline{V}(O_i) = V(O_i) \bigcup \partial V(O_i)$. The *Voronoi diagram* of the workspace is defined as the union of all cell boundaries [58] i.e.

$$D(W) = \bigcup_{O_i \in S_O} \partial V(O_i). \tag{68}$$

The shared boundary between two Voronoi cells is a *Voronoi edge*. More specifically, a Voronoi edge between two Voronoi cells $V(O_i)$ and $V(O_j)$ is defined by

$$E_{ij} = \partial V(O_i) \bigcap \partial V(O_j). \tag{69}$$

3.2 Tracking a Voronoi Edge

In this section, we develop a feedback control law to make a vehicle, with its dynamics approximated by a unit speed particle, move along a Voronoi edge. We assume that the range sensors of the vehicle can detect two obstacle boundaries, each on one side of the vehicle. Then, on each obstacle boundary, the vehicle can determine the closest point to itself. The feedback control law uses measurements at the two closest points.

We first introduce the shape dynamics that govern the relationship between the vehicle and the closest points. We then derive the control law, and prove its convergence.

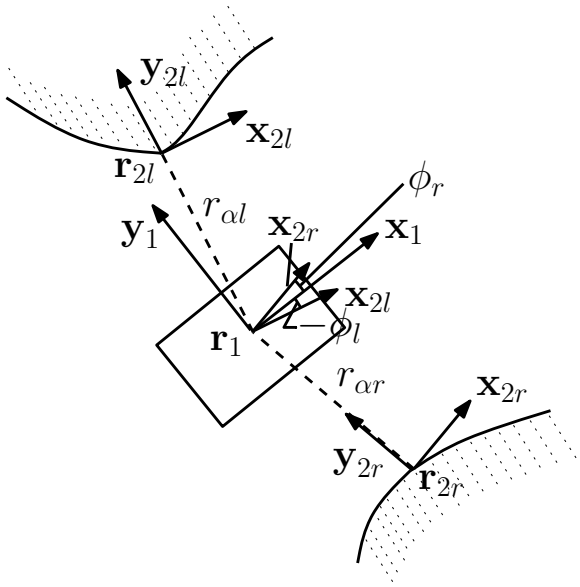


Figure 18: A vehicle with obstacle boundaries to both the left and the right hand sides of the vehicle.

3.2.1 Shape Dynamics

In Figure 18, $\mathbf{r}_1$ denotes the position of the vehicle, and $\mathbf{x}_1$ denotes the heading direction of the vehicle. $\mathbf{r}_{2r}$ is the closest point to the right of the vehicle, and $\mathbf{x}_{2r}$ denotes the unit tangent vector to the boundary curve at $\mathbf{r}_{2r}$. ϕ_r is the angle measured counter-clockwise from $\mathbf{x}_1$ to $\mathbf{x}_{2r}$, the tangent vector at $\mathbf{r}_{2r}$. The relative position between the vehicle and the closest point to the right of the vehicle is $\mathbf{r}_{\alpha r} = \mathbf{r}_{2r} - \mathbf{r}_1$, and we define $r_{\alpha r} = \|\mathbf{r}_{\alpha r}\|$.

The quantities $\mathbf{r}_{2l}$, $\mathbf{x}_{2l}$, ϕ_l, $\mathbf{r}_{\alpha l}$, and $r_{\alpha l}$ are defined to the left of the vehicle in the same fashion as those to the right of the vehicle.

We choose the positive directions of the boundary curves such that

$$\begin{aligned} \mathbf{x}_1 \cdot \mathbf{x}_{2l} &= \cos(\phi_l) > 0 \\ \mathbf{x}_1 \cdot \mathbf{x}_{2r} &= \cos(\phi_r) > 0, \end{aligned} \tag{70}$$

which means that $-\pi/2 < \phi_l < \pi/2$ and $-\pi/2 < \phi_r < \pi/2$.

Considering the boundary curve to the right of the vehicle, the shape dynamics

are given by [89] as follows.

$$\dot{r}_{\alpha r} = -\sin(\phi_r) \tag{71}$$

$$\dot{\phi}_r = (\frac{\kappa_r}{1-\kappa_r r_{\alpha r}})\cos(\phi_r) - u, \tag{72}$$

where κ_r denotes the algebraic curvature of the boundary at the closest point to the right of the vehicle. Similarly, for the boundary curve to the left, we have

$$\dot{r}_{\alpha l} = \sin(\phi_l) \tag{73}$$

$$\dot{\phi}_l = (\frac{\kappa_l}{1+\kappa_l r_{\alpha l}})\cos(\phi_l) - u, \tag{74}$$

where κ_l denotes the algebraic curvature of the boundary at the closest point to the left of the vehicle.

3.2.2 Tracking Control and Convergence Analysis

In this section, we design the control law based on a Lyapunov function. Consider the Lyapunov function candidate

$$V = -ln(\cos(\frac{\phi_l+\phi_r}{2})) + \lambda(r_{\alpha l} - r_{\alpha r})^2, \tag{75}$$

where $\lambda > 0$ is a constant that balances the control for alignment and the control for vehicle position. In (75), the term $-ln(\cos(\frac{\phi_l+\phi_r}{2}))$ penalizes misalignment between the heading direction of the vehicle and the tangent vector to the Voronoi edge. The term $r_{\alpha l} - r_{\alpha r}$ in (75) makes the vehicle converge to a Voronoi edge. The time derivative of V is

$$\begin{aligned}\dot{V} = \tan(\frac{\phi_l+\phi_r}{2})[&\frac{1}{2}(\frac{\kappa_l\cos\phi_l}{1+\kappa_l r_{\alpha l}} + \frac{\kappa_r\cos\phi_r}{1-\kappa_r r_{\alpha r}} - 2u) \\ &+4\lambda(r_{\alpha l} - r_{\alpha r})\cos(\frac{\phi_l+\phi_r}{2})\cos(\frac{\phi_l-\phi_r}{2})],\end{aligned} \tag{76}$$

where we have used the shape dynamics (71),(72),(73), and (74). Also, $\sin(\phi_l) + \sin(\phi_r) = 2\sin(\frac{\phi_l+\phi_r}{2})\cos(\frac{\phi_l-\phi_r}{2})$ is applied.

We design the steering control u so that $\dot{V} \leq 0$. One choice of u that leads to $\dot{V} \leq 0$ is

$$u = \frac{1}{2}\left(\frac{\kappa_l \cos\phi_l}{1+\kappa_l r_{\alpha l}} + \frac{\kappa_r \cos\phi_r}{1-\kappa_r r_{\alpha r}}\right) + \mu \sin\left(\frac{\phi_l + \phi_r}{2}\right) + 2\lambda(r_{\alpha l} - r_{\alpha r})(\cos\phi_l + \cos\phi_r), \quad (77)$$

where $\mu > 0$ is a constant gain for the tracking controller. According to [89], we see that the steering control u corresponds to the curvature of the vehicle's trajectory at the moment when u is applied. Hence, as long as the control law (77) is not singular (denominator of (77) is not zero), curvature is well-defined along the trajectory of the vehicle. Hence, we can guarantee that the trajectory of the vehicle is smooth.

The time derivative of V in (76) with u given by (77) is

$$\dot{V} = -\mu \frac{\sin^2\left(\frac{\phi_l+\phi_r}{2}\right)}{\cos\left(\frac{\phi_l+\phi_r}{2}\right)}. \quad (78)$$

By letting $\phi = \frac{\phi_l+\phi_r}{2}$, we get $\dot{V} = -\mu\frac{\sin^2(\phi)}{\cos(\phi)}$. In addition, $-\pi/2 < \phi < \pi/2$ is derived using (70). Within this range, we obtain $-\infty < \dot{V} \leq 0$. $\dot{V} = 0$ if and only if $\phi = 0$, since $-\pi/2 < \phi < \pi/2$. As $|\phi|$ increases from 0 to $\pi/2$, $\dot{V}$ monotonically decreases to $-\infty$. This is to penalize misalignment between the heading direction of the vehicle and the tangent vector to the Voronoi edge.

Theorem 3. *Suppose that* $1 + \kappa_l r_{\alpha l} \neq 0$ *and that* $1 - \kappa_r r_{\alpha r} \neq 0$. *Then, using the steering control law in (77), the unit speed vehicle, whose initial position satisfies (70), converges to the state where it moves along a smooth Voronoi edge.*

Proof. For each trajectory that initially satisfies (70), there exists a compact sublevel set Ω of V such that the trajectory remains in Ω for all future time. Then, by LaSalle's Invariance Principle [47], the trajectory converges to the largest invariant set I within the set E that contains all points in Ω where $\dot{V} = 0$. The set E in this case is the set of all points in Ω such that $\phi_l + \phi_r = 0$. Thus, at any point in E, the dynamics are expressed as

$$E = \{(r_{\alpha l}, r_{\alpha r}, \phi_l, \phi_r) | \phi_l + \phi_r = 0\}. \quad (79)$$

Since the trajectory converges to the largest invariant set I within the set E, we obtain $\dot{\phi}_l + \dot{\phi}_r = 0$ in I. Therefore, using (72) and (74), we derive

$$(\frac{\kappa_r}{1-\kappa_r r_{\alpha r}})\cos(\phi_r) + (\frac{\kappa_l}{1+\kappa_l r_{\alpha l}})\cos(\phi_l) - 2u = 0. \tag{80}$$

Applying (77), we get

$$2\lambda(r_{\alpha l} - r_{\alpha r})(\cos(\phi_l) + \cos(\phi_r)) + \mu \sin(\frac{\phi_l + \phi_r}{2}) = 0. \tag{81}$$

In order to satisfy (81), $r_{\alpha l} - r_{\alpha r} = 0$ is required, since $\phi_l + \phi_r = 0$ inside the set E. Therefore, the largest invariant set I is expressed as

$$I = \{(r_{\alpha l}, r_{\alpha r}, \phi_l, \phi_r) | r_{\alpha l} = r_{\alpha r}, \phi_l + \phi_r = 0\}. \tag{82}$$

Thus, we can conclude that $(r_{\alpha l}, r_{\alpha r}, \phi_l, \phi_r)$ converges to the equilibrium where $r_{\alpha l} = r_{\alpha r}$ and $\phi_l = -\phi_r$. If the vehicle is equidistant from two closest points on the obstacle boundaries and the heading direction of the vehicle is aligned to the tangent vector to a Voronoi edge, then the vehicle moves along the Voronoi edge. This implies that, as the vehicle converges to the state I in (82), it converges to move along a Voronoi edge.

□

By means of the LaSalle's Invariance Principle, we can conclude asymptotic convergence. This may cause a problem for a vehicle to track a Voronoi edge with finite length. This problem can be alleviated by noticing that the convergence rate of the control law depends on the controller gain μ (see (78)). Larger gain μ will enable the vehicle to converge to a Voronoi edge faster, which has been confirmed by rigorously computing the eigenvalues of the Jacobian matrix for linearized closed loop dynamics near the tracking equilibrium. If a lower bound of the length of a Voronoi edge within the workspace is known, then μ can be selected so that the vehicle gets sufficiently close to the Voronoi edge in finite time. Such a lower bound can be estimated based

on the length of the Voronoi edges already detected by the vehicle, which will result in an adaptive gain μ. The details of the gain adjustment algorithm is not the main focus of this chapter.

3.3 The Boundary Expansion (BE) Algorithms

In this section, we propose the boundary expansion (BE) algorithms that enable the vehicle to construct the Voronoi diagram of W by traversing all Voronoi edges E_{ij} for $i, j = 1, 2, ..., M$.

3.3.1 Definitions and Assumptions

We define an *intersection* P as a point at which the following conditions are satisfied:

- there exists a circle centered at P intersecting obstacle boundaries at more than two points. These points on obstacle boundaries are called the *closest points* at the intersection. If the vehicle is at an intersection, then the closest points correspond to the points that have local minimal distances to the vehicle.
- the interior of the circle does not intersect any obstacles.

The circle centered at P satisfying the above conditions is called an *intersection circle* (see Figure 19). The lines connecting the intersection and the closest points on the obstacle boundaries partition the intersection circle into *sector*s. We can see that each sector is the "pie-shaped area" within the intersection circle (see Figure 19).

Suppose that the vehicle under control moves along E_{ij} until it visits an intersection P, as illustrated in Figure 19. It will detect two closest points on ∂O_i and ∂O_j, since $P \in E_{ij}$. The sector that has these two closest points as its end points is defined as *sector* 0 for the intersection P. Intuitively, sector 0 is the sector through which the vehicle moves to reach the intersection P. It serves as a starting point for indexing the rest of the sectors. Suppose that there are n sectors in the intersection

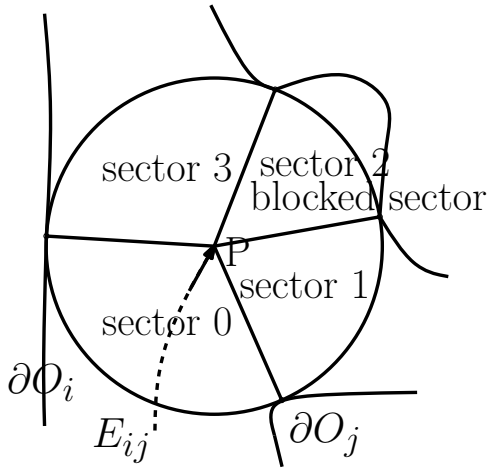


Figure 19: The vehicle at the intersection P. The circle is the intersection circle. The closest points partition the circle into sectors. For $i = 1, 2, 3$, the sector i is adjacent to the sector $i - 1$ in the counter-clockwise direction.

circle, as seen in Figure 19. Looking into the page, we then index the sectors in the counter-clockwise direction from sector 0. The index k satisfies $0 \leq k \leq n - 1$.

When two end points of a particular sector are on the same obstacle, the sector is called a *blocked sector*, which is illustrated as "sector 2" in Figure 19. An *open sector*, illustrated as "sector 1" and "sector 3" in Figure 19, denotes a sector that is neither a blocked sector nor a sector 0. If the intersection detected by the vehicle has an open sector that has not been visited by the vehicle, then the intersection is marked as *unexplored*. Otherwise, the intersection is marked as *explored*.

The following assumptions are made about the workspace and the capability of the vehicle.

(A1) $\partial V(O_i)$ is a simple closed curve for each $O_i \in S_O$. In other words, $\partial V(O_i)$ is continuous and no self-intersection occurs.

(A2) All blocked sectors for every intersection are detectable by a vehicle[1].

(A3) $\bigcup_{O_i \in S_O} \overline{V}(O_i) = W$, where $\overline{V}(O_i) = V(O_i) \bigcup \partial V(O_i)$.

[1]The experiments in Section 3.6.2 verify that the robot can detect a blocked sector using IR sensors.

(A4) A vehicle can distinguish O_M from other obstacles[2]. The initial position of a vehicle is such that an obstacle other than O_M is detected to the right of the vehicle[3].

We call a simple closed curve that contains intersections connected by Voronoi edges an *enclosing boundary* if there is no unexplored intersection strictly inside such a curve. At any moment in the BE algorithms, the enclosing boundary is unique.

3.3.2 Data Structures

The data structures used in the BE algorithms are summarized in Table 1. For each intersection detected by the vehicle, we store the coordinate of the intersection. The enclosing boundary can then be represented by a circularly linked list L constructed by linking the intersections.

We use a graph structure G to represent the Voronoi diagram under construction. G contains a list of distinct intersections together with an adjacency matrix whose entries indicate whether a particular edge is in the graph. When the vehicle detects an intersection P that has not been stored in G, then the adjacency matrix is expanded to include P.

3.3.3 Initialize the Enclosing Boundary

Algorithm 1 is to initialize the enclosing boundary. Suppose that the obstacle to the right of the vehicle is O_i. Under the control law, the vehicle converges to the state that it moves along $\partial V(O_i)$ with ∂O_i to its right. We denote the first intersection on $\partial V(O_i)$ that the vehicle encounters as $P_{1,0}$. At each intersection that the vehicle encounters, it searches for an open sector in the counter-clockwise direction, from the reader's view, from sector 0. Once an open sector is detected, the vehicle moves

[2]We can consider specific sensors deployed along O_M so that the vehicle can distinguish O_M from other obstacles. Or, O_M may have a different shape (or color) from other obstacles.

[3]This is strictly speaking not a restriction, since the vehicle can initialize the heading orientation so that an obstacle other than O_M is detected to the right of the vehicle.

Table 1: Table of Data Structures and Operations
L_u: singly linked list representing the enclosing boundary under construction.
L_u.Insert(P): insert an intersection P at the end of the linked list L_u.
L: circularly linked list representing the current enclosing boundary.
HT=L.seg(*head*,*tail*): segment of L that starts from the *head* and ends at the *tail*.
L_r= L.Remove(HT): remove the segment HT from L resulting in L_r.
CS: singly linked list representing the candidate segment.
L=L_r.Combine(CS): combine the linked list L_r with CS resulting in updated L.
G: graph structure, representing the Voronoi diagram under construction, which contains a list of intersections and an adjacency matrix.
G.Update(P): update entries of the adjacency matrix associated to an intersection P.
G.Expand(P): expand the adjacency matrix to include an intersection P, and update entries of the matrix associated to P.
HT.Search(*unexplored*): search for unexplored intersections in the linked list HT. If there is no unexplored intersection, return $NULL$.
D_k: disabled intersection set of B_k. Here, B_k denotes the enclosing boundary updated after k steps.
D_k.Store(P): store an intersection P in D_k.

through the sector. Iterating this, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right and a sequence of intersections encountered along its path is constructed. The initial enclosing boundary B_0 is defined as the sequence of Voronoi edges connecting this intersection sequence until the vehicle is at $P_{1,0}$ for the second time.

3.3.4 Update the Enclosing Boundary

Let B_k denote the enclosing boundary updated after k steps. Algorithm 2 will expand B_0 to obtain B_k for $k = 1, 2, ...$ until B_k encloses all obstacles except for O_M. We expand the enclosing boundary while maintaining it as a simple closed curve tracked by the vehicle in the clockwise direction.

The boundary expansion is guaranteed by two rules, called the *sector selection rules*, that decide which sector the vehicle should move through at an intersection and when to update the enclosing boundary.

Before stating the sector selection rules, we introduce the *(pointer) sector* and the *(pointer + 1) sector*. When the vehicle on the enclosing boundary leaves for the

Algorithm 1 Construct the Initial Enclosing Boundary

$i \leftarrow 1$;
repeat
  the vehicle encounters an intersection;
  $P_{i,0} \leftarrow$ the intersection;
  search for an open sector in the counter-clockwise direction from sector 0; the vehicle moves through the first open sector;
  $P_{i,0}$.pointer $\leftarrow$ first open sector;

  if $P_{i,0}$ has an open sector that has not been visited by the vehicle **then**
    $P_{i,0}$.mark $\leftarrow$ *unexplored*;
  else
    $P_{i,0}$.mark $\leftarrow$ *explored*;
  end if
  L_u.Insert($P_{i,0}$);
  if $P_{i,0} \in G$ **then**
    G.Update($P_{i,0}$);
  else
    G.Expand($P_{i,0}$);
  end if
  $i \leftarrow i + 1$;
until the vehicle encounters $P_{1,0}$ for the second time;
$L \leftarrow L_u$;

next intersection along the enclosing boundary in the clockwise direction, it must move through another sector that contains the path leading to the next intersection. We call this sector the (*pointer*) sector. The (*pointer* + 1) sector denotes the sector whose index is larger than the (*pointer*) sector by one. The (*pointer*) sector and the (*pointer* + 1) sector stored at every intersection on the enclosing boundary are illustrated in Figure 20.

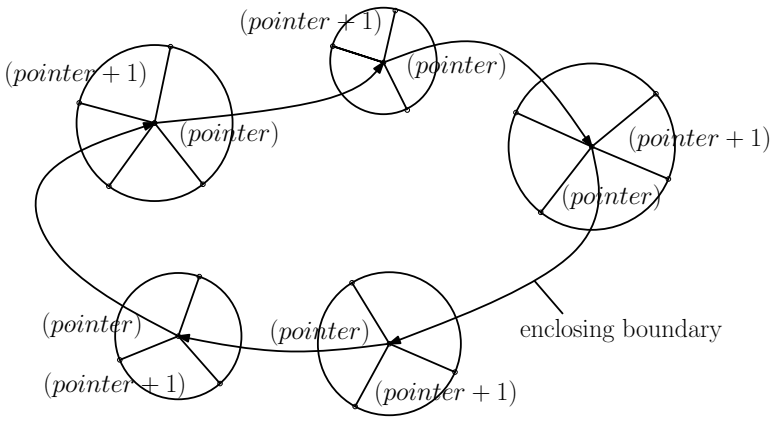


Figure 20: The (*pointer*) sector and the (*pointer* + 1) sector stored at every intersection on the enclosing boundary.

The sector selection rules are stated for two cases:

R1 When the vehicle visits an intersection on the enclosing boundary, the vehicle searches for an open sector in the counter-clockwise direction from the (*pointer*+ 1) sector to sector 0. Once an open sector is detected, the following condition is checked. If the vehicle would move through the open sector, then O_M would not lie to the right of the vehicle. If an open sector is detected that satisfies this condition, the vehicle moves through the open sector. Otherwise, the vehicle moves through the (*pointer*) sector.

R2 When the vehicle visits an intersection not on the enclosing boundary, the vehicle searches for an open sector in the counter-clockwise direction from sector 0. Once an open sector is detected, the vehicle moves through the open sector.

Suppose that the vehicle is on the Voronoi edge $E_{ij} \subset \partial V(O_i)$, where $i \neq j$. At any intersection on $\partial V(O_i)$, there exist two sectors that lead the vehicle to follow $\partial V(O_i)$ in the clockwise or in the counter-clockwise direction. Therefore, the vehicle can always find an open sector that satisfies the sector selection rule R2.

Under the sector selection rules, the behavior of the vehicle is as follows. The vehicle moves along the enclosing boundary until it visits an intersection where there is an open sector that leads outside the enclosing boundary but will not force the vehicle to track O_M to its right. Then, the vehicle marks the intersection as *head* and moves through the open sector. A singly linked list CS is initiated with the *head*. Thereafter, the vehicle keeps moving and chooses sectors using the rule R2, inserting all intersections it encounters into CS. This process ends when the vehicle encounters the enclosing boundary again at an intersection. The vehicle marks this intersection as *tail* and inserts *tail* into CS. We call the trajectory of the vehicle from the *head* to the *tail* the *candidate segment*. After the vehicle gets to the *tail*, it uses the rule R1 to determine which sector to move through.

We introduce the *boundary updating rule*. This rule regulates when to replace a segment of the current enclosing boundary with the candidate segment CS. The rule is as follows:

R3 If there is no unexplored intersection, strictly between the *head* and the *tail*, along the segment of enclosing boundary in the clockwise direction, then we replace the segment of enclosing boundary from the *head* to the *tail* by the candidate segment.

Suppose the current enclosing boundary is B_k. Figure 21 illustrates the case where the boundary updating rule is satisfied. In this case, we update B_k by replacing the segment of enclosing boundary that starts from the *head* and ends at the *tail* by the candidate segment. Figure 22 shows the update of L, the data structure of the enclosing boundary, when the boundary updating rule is satisfied.

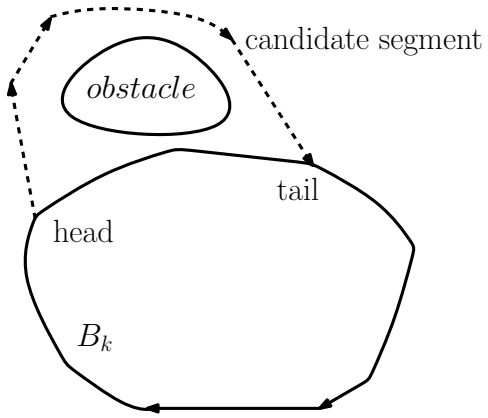


Figure 21: The case where the enclosing boundary can be expanded.

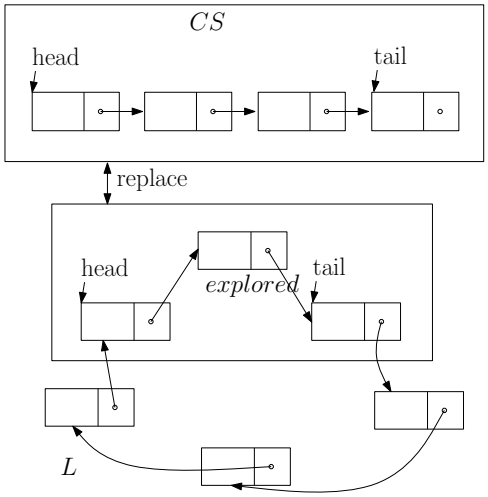


Figure 22: The update of the circularly linked list associated with boundary expansion.

Figure 23 illustrates the case where the boundary updating rule is not satisfied. The dotted curve indicates the unexplored Voronoi edge. There are two unexplored intersections along the segment of enclosing boundary from the *head* to the *tail*. If the rule for updating B_k is not satisfied, as illustrated in Figure 23, then we keep the enclosing boundary unchanged. To prevent the vehicle from repeatedly traversing the candidate segment that does not lead to boundary updates, the *head* of such a candidate segment is recorded as a *disabled intersection* in a set D_k that is associated with B_k. If the vehicle encounters a disabled intersection, it will ignore this intersection and move along B_k to the next intersection.

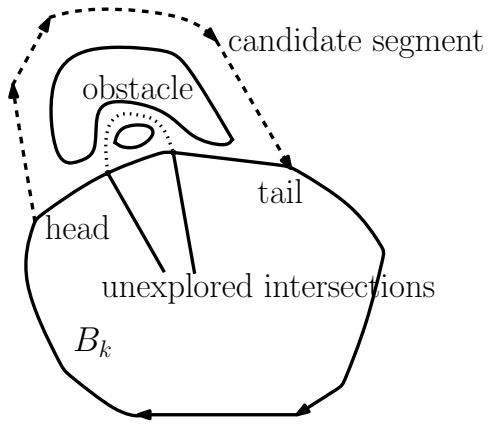


Figure 23: The case where boundary expansion is not performed according to the boundary updating rule R3.

3.4 Convergence of the BE algorithms

In this section, we prove the convergence of the boundary expansion algorithms, i.e., both algorithm 1 and algorithm 2.

Lemma 1. *Consider the vehicle and the workspace W satisfying assumptions (A1)-(A4). Suppose that the obstacle to the right of the vehicle is O_i. Then, using algorithm 1, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right and a sequence of intersections encountered along its path is constructed. Algorithm 1 terminates when the vehicle returns to the first intersection in the sequence.*

Algorithm 2 Boundary Expansion

N denotes the number of intersections on L; label the intersections on L in the clockwise direction as $P_{1,0}$,$P_{2,0}$,...,$P_{N,0}$; $i \leftarrow 1$ and $k \leftarrow 0$;
while there is an obstacle other than O_M outside the enclosing boundary **do**
  the vehicle visits $P_{i,k}$ on L;
  if $P_{i,k} \notin D_k$ and there exists an open sector, satisfying the sector selection rule R1, outside the enclosing boundary **then**
    $m \leftarrow 1$; $E_1 \leftarrow P_{i,k}$; $MeetTail \leftarrow 0$;
    while $MeetTail \neq 1$ **do**
      the vehicle finds E_m;
      if $m == 1$ **then**
        move through the open sector selected using the rule R1;
      else
        search for an open sector satisfying the rule R2, and move through the selected open sector;
      end if
      E_m.pointer$\leftarrow$ the selected sector;
      if E_m has an open sector that has not been visited by the vehicle **then**
        E_m.mark $\leftarrow$ *unexplored*;
      else
        E_m.mark$\leftarrow$ *explored*;
      end if
      CS.Insert(E_m);
      if $E_m \in G$ **then**
        G.Update(E_m);
      else
        G.Expand(E_m);
      end if
      if $m \neq 1$ and $E_m == P_{T,k}$ for any T **then**
        $MeetTail \leftarrow 1$;
      else
        $m \leftarrow m + 1$;
      end if
    end while
    $head \leftarrow E_1$; $tail \leftarrow P_{T,k}$; HT=L.seg($head$,$tail$);
    if $HT \neq NULL$ and HT.Search($unexplored$)$\subset$ ($head$, $tail$) **then**
      L_r=L.Remove(HT); L=L_r.Combine(CS); N denotes the number of intersections on L; $P_{1,k+1} \leftarrow tail$; relabel the intersections on L in the clockwise direction as $P_{1,k+1}$,$P_{2,k+1}$,...,$P_{N,k+1}$; $i \leftarrow 1$; $k \leftarrow k + 1$;
    else
      D_k.Store($head$); $i \leftarrow T$;
    end if
  else
    $i \leftarrow i + 1$;
  end if
  if $i > N$ **then**
    $i \leftarrow i - N$;
  end if
end while

Proof. Suppose that the obstacle to the right of the vehicle is O_i. Under the control law, the vehicle converges to the state where it moves along $\partial V(O_i)$ with ∂O_i to its right. We denote the first intersection on $\partial V(O_i)$ that the vehicle encounters as $P_{1,0}$, and label the intersections the vehicle will encounter if it follows $\partial V(O_i)$ with ∂O_i to its right as $(P_{1,0}, P_{2,0}, ..., P_{N,0})$. We organize our proofs in two steps:

1. Show that the vehicle moves to $P_{2,0}$.

2. Show that the vehicle visits $P_{2,0} \rightarrow P_{3,0}... \rightarrow P_{N,0} \rightarrow P_{1,0}$.

1. For convenience, we call the closest point on ∂O_i as $C_{\partial O_i}$. At $P_{1,0}$, $C_{\partial O_i}$ is to the right of the vehicle. Sector 0 and sector 1 have the common closest point at $C_{\partial O_i}$. The vehicle moves through sector 1 if it is not blocked. If sector 1 is blocked, then the sector selection rule R2 is applied so that the vehicle moves through the next open sector having $C_{\partial O_i}$ to the right of the vehicle. Therefore, the vehicle tracks $\partial V(O_i)$ with ∂O_i to its right and will encounter $P_{2,0}$.
2. Consider the case where the vehicle visits $P_{k,0}$ starting from $P_{k-1,0}$ for all $2 \leq k \leq N$. Similar to step 1, using the sector selection rule R2, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right until it visits $P_{k+1,0}$.

By induction, if the vehicle uses the sector selection rule R2 at $P_{1,0}, P_{2,0}, ...P_{N,0}$, then it visits the intersections following the sequence $P_{1,0} \rightarrow P_{2,0}... \rightarrow P_{N,0} \rightarrow P_{1,0}$. Algorithm 1 ends when the vehicle returns to $P_{1,0}$.

□

To state Theorem 4, we need to introduce a few new notations: Let Q denote an obstacle, other than O_M, outside B_k such that $B_k \bigcap V(Q) \neq \emptyset$. If Q is such that $B_k \bigcap V(Q)$ is a connected edge segment of B_k, then we call it an *addable obstacle* Q^k. Other than this possibility, based on the assumption that the boundary of every Voronoi cell is a simple closed curve, there are only two more possibilities that Q can have. Let Q^t denote an obstacle that $B_k \bigcap V(Q^t)$ is an intersection. Q^d denotes an

obstacle such that $B_k \bigcap V(Q^d)$ is composed of disjoint edge segments or intersections of B_k. $V(Q^t), V(Q^d)$, and $V(Q^k)$ are illustrated in Figure 24.

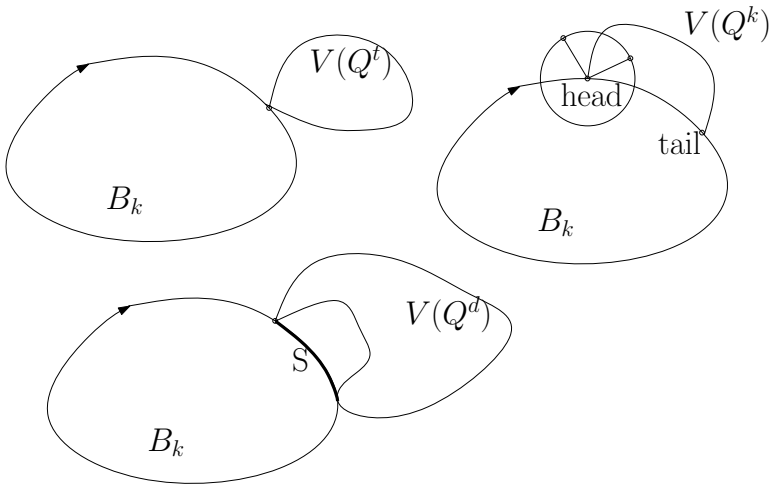


Figure 24: Illustration of $V(Q^t), V(Q^d), V(Q^k)$, and S. Q^k is addable but Q^d and Q^t are not addable.

Theorem 4. *Consider the vehicle and the workspace W satisfying assumptions (A1)-(A4). The vehicle explores W using algorithm 2. As long as there exists an obstacle other than O_M outside the enclosing boundary B_k, the following assertions hold:*

1. *B_k is a simple closed curve traversed in the clockwise direction, and there is no unexplored intersection strictly inside B_k.*

2. *There exists an addable obstacle Q^k such that the vehicle will move along a path $CS \subset \partial V(Q^k)$, but $CS \neq \partial V(Q^k) \bigcap B_k$. The path CS intersects B_k at two intersections that can be marked as head and tail. Furthermore, CS is the candidate segment satisfying the rule R3 for updating B_k.*

3. *B_k will be expanded so that the obstacle Q^k will be inside the expanded enclosing boundary B_{k+1}.*

Proof. Using algorithm 1, the vehicle moves along $\partial V(O_i)$ with ∂O_i to its right and a sequence of intersections encountered along its path is constructed according to

Lemma 1. Therefore, B_0 is in the clockwise direction from the reader's viewpoint, which is identical to $\partial V(O_i)$. Here, $B_0 = \partial V(O_i)$ is a simple closed curve using assumption (A1). Furthermore, no intersection is strictly inside B_0.

We prove by induction. Suppose that B_k is a simple closed curve in the clockwise direction and that there exists an obstacle other than O_M outside B_k. Suppose that there is no unexplored intersection strictly inside B_k. Now, we organize our proofs in four steps:

1. show that there exists an addable obstacle Q^k as long as there exists an obstacle other than O_M outside B_k.

2. show that there exists no unexplored intersection strictly between the starting intersection (*head*) and the ending intersection (*tail*) of $\partial V(Q^k) \bigcap B_k$.

3. show that the vehicle moves along the path $CS \subset \partial V(Q^k)$ and that the path intersects B_k at the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$.

4. show that, after new enclosing boundary B_{k+1} is generated, Q^k is inside B_{k+1}. B_{k+1} is a simple closed curve traversed by the vehicle in the clockwise direction, and there is no unexplored intersection strictly inside B_{k+1}.

1. First, we show that there exists Q as long as there is an obstacle other than O_M outside B_k. Suppose that all obstacles O_i outside B_k are such that $B_k \bigcap V(O_i) = \emptyset$. In this case, $\partial V(O_M)$ cannot be a simple closed curve, since $\partial V(O_M)$ should enclose both B_k and O_i.

Next, we prove the existence of Q^k by contradiction. Suppose all Q are either Q^d or Q^t. We first argue that Q^d must exist. If only Q^t exists, then $\partial V(O_M)$ has self-intersection, since $\partial V(O_M)$ should enclose both B_k and Q^t. For Q^d, call the disjoint boundary segments as $B_k \bigcap V(Q^d)$. Along B_k, there exist one or more edge segments of B_k connecting these disjoint boundary segments. We select one segment S such

that S and some edges of $\partial V(Q^d)$ form a closed loop that does not enclose Q^d. S is illustrated in Figure 24. This closed loop can be constructed as a simple closed curve, since no self-intersection occurs along $\partial V(Q^d)$, as well as along $S \subset B_k$. We call this closed loop Q_1^d. Inside the closed loop Q_i^d, we iteratively find another loop for Q_{i+1}^d until no more Q_{i+1}^d exists. Voronoi edges in S that exist along the inner most loop belong to neither $V(Q^d)$ nor $V(Q^t)$ for any Q^t, which implies that there exists an addable obstacle Q^k inside the inner most loop. Therefore, by contradiction, there exists an addable obstacle Q^k as long as there exists an obstacle other than O_M outside B_k.

2. We prove by contradiction. Suppose that an unexplored intersection exists strictly between the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$. Then, there exists an unvisited Voronoi edge meeting the unexplored intersection. Since we suppose that no unexplored intersection is strictly inside B_k, this unvisited Voronoi edge leads outside of the enclosing boundary toward Q^k, as illustrated in Figure 25. Hence, at this unexplored intersection, three edges of $\partial V(Q^k)$ meet, resulting in self-intersection of $\partial V(Q^k)$. This is a contradiction to assumption (A1).

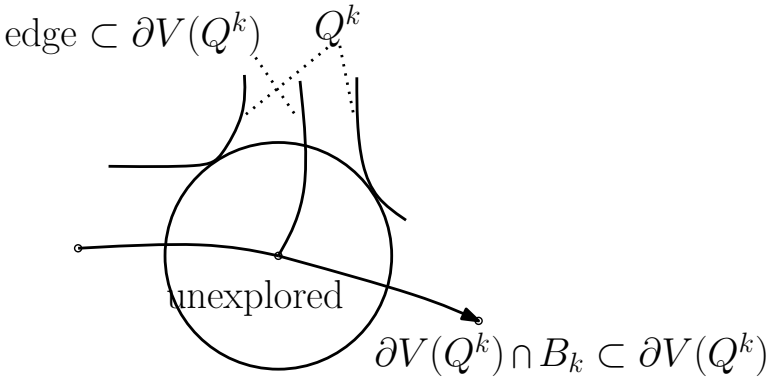


Figure 25: Three edges of $\partial V(Q^k)$ meeting at an unexplored intersection on $\partial V(Q^k) \bigcap B_k$. This is impossible, since we assume that the boundary of each Voronoi cell is a simple closed curve.

3. We suppose that the vehicle has tracked B_k in the clockwise direction until it visits the starting intersection of $\partial V(Q^k) \bigcap B_k$. Then, we mark the starting intersection as

head and mark the ending intersection of $\partial V(Q^k) \bigcap B_k$ as *tail*. Note that the direction of $\partial V(Q^k) \bigcap B_k$ is from the *head* to the *tail*, since B_k is in the clockwise direction.

When the vehicle visits the *head*, there exists an open sector outside the enclosing boundary, as illustrated in Figure 24. Then, according to the rule R1, the vehicle moves through the open sector outside the enclosing boundary with Q^k to the vehicle's right. Thereafter, it chooses sectors using the rule R2 and moves along Voronoi edges.

We label the intersections the vehicle encounters if it follows $\partial V(Q^k)$ with Q^k to its right as $(E_1 = head, E_2, ..., E_n = tail)$. Similar to the proof of Lemma 1, the vehicle starting from E_m moves along $\partial V(Q^k)$ with Q^k to its right until it visits E_{m+1}. The sequence of Voronoi edges connecting the intersection sequence $(E_1 = head, E_2, ..., E_n = tail)$ is defined as the candidate segment $CS \subset \partial V(Q^k)$.

4. The boundary updating rule for B_k is satisfied. Thus, we update B_k by substituting $\partial V(Q^k) \bigcap B_k$ for CS. There is no unexplored intersection strictly inside B_{k+1}, because there exists no unexplored intersection strictly between the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$.

We now prove that B_{k+1} is a simple closed curve in the clockwise direction. Since self-intersection of CS cannot occur as we substitute $\partial V(Q^k) \bigcap B_k$ for CS, B_{k+1} is a simple closed curve. In addition, the direction of B_{k+1} is in the clockwise direction, since the direction of $\partial V(Q^k) \bigcap B_k$ is the same as that of CS.

Next, since $CS \bigcup (\partial V(Q^k) \bigcap B_k) \subset \partial V(Q^k)$ is a simple closed curve, $CS \bigcup (\partial V(Q^k) \bigcap B_k) = \partial V(Q^k)$. After we generate B_{k+1}, the region enclosed by $CS \bigcup (\partial V(Q^k) \bigcap B_k)$ is inside B_{k+1}, i.e., Q^k is inside B_{k+1}. We have proved all statements in Theorem 4.

□

Corollary 1. *Under algorithms 1 and 2, the enclosing boundary converges in finite time to the state that there is no obstacle other than O_M outside the enclosing boundary.*

Proof. As long as there is an obstacle other than O_M outside B_k, we can generate B_{k+1} using Theorem 4. The process ends when there is no obstacle other than O_M outside B_k. Since there are finite number of obstacles, the process terminates in finite time.

□

Corollary 2. *When algorithm 2 terminates, a complete Voronoi diagram is constructed for W.*

Proof. When algorithm 2 terminates, there is no unexplored intersection outside the final enclosing boundary, since there is only O_M outside. And, according to Theorem 4, there is no unexplored intersection strictly inside the enclosing boundary either. Thus, all intersections in W are explored. This implies that all Voronoi edges in W are visited by the vehicle. Consequently, the trajectory of the vehicle depicts all Voronoi edges in W and a complete Voronoi diagram is constructed.

□

3.5 Performance Analysis

In this section, we provide an analytical upper bound for the total time spent to construct the Voronoi diagram in a regularized workspace. As the number of Voronoi cells in a bounded workspace increases, each Voronoi cell approaches to hexagonal shape [31, 74]. Thus, we analyze the performance of the BE algorithms in the workspace where each cell has a hexagonal shape with identical size, as illustrated in Figure 26. In the appendix, we state the conditions for a workspace to obtain hexagonal Voronoi cells.

Theorem 5. *Consider a unit speed vehicle and a workspace W with assumptions (A1)-(A4) satisfied. Suppose that there are M obstacles such that all obstacles, except for O_M, have hexagonal Voronoi cells with identical size. Using the BE algorithms,*

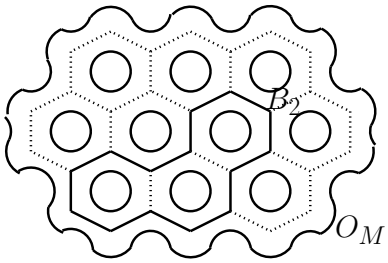


Figure 26: A workspace where all Voronoi cells, except for $V(O_M)$, have hexagonal shapes with identical size. Inside B_2, there are 3 obstacles.

the exploration time is bounded above by $T(\frac{1}{2}M^2 - \frac{1}{2}M)$ *where* T *denotes the time for the vehicle to traverse along the edges of one hexagonal Voronoi cell.*

Proof. Consider the time to build B_0. Since there is only one Voronoi cell inside B_0, the time to construct B_0 is

$$T_{B_0} = T. \tag{83}$$

Next, consider the time to generate B_{k+1} from B_k where $k \geq 0$. Suppose that B_k is generated and that the vehicle is at the *tail* of the candidate segment (CS) for generating B_k.

Using Theorem 4, at least one Voronoi cell, which is outside B_k and intersects the perimeter of B_k, is the Voronoi cell for an addable obstacle Q^k. This addable obstacle Q^k will be an obstacle that is inside B_{k+1}. The vehicle moves along B_k to reach the starting intersection (*head*) of $\partial V(Q^k) \bigcap B_k$. The vehicle's maximal traversal distance to meet the starting intersection of $\partial V(Q^k) \bigcap B_k$ is bounded above by the length of B_k. Note that the vehicle has unit speed and that the number of Voronoi cells, which are inside B_k, is $k+1$. Therefore, the length of B_k is bounded above by $(k+1)T$. Furthermore, the length of CS, which connects the starting and the ending intersections of $\partial V(Q^k) \bigcap B_k$, is bounded above by T. Hence, we derive the time to construct B_{k+1} as

$$T_{B_{k+1}} \leq T_{B_k} + (k+1)T + T, \tag{84}$$

where T_{B_k} denotes the time to construct B_k. Using (84), we obtain

$$T_{B_k} \leq T(\tfrac{1}{2}k^2 + \tfrac{3}{2}k + 1), \tag{85}$$

since $T_{B_0} = T$ (see (83)). There are $k+1$ and $M-1$ obstacles inside B_k and $\partial V(O_M)$ respectively. Therefore, our algorithms terminate when

$$k + 1 = M - 1. \tag{86}$$

Hence, replacing k in (85) by $M-2$, we obtain the upper bound on time for the construction of the Voronoi diagram using the BE algorithms as

$$T_c \leq T(\frac{1}{2}M^2 - \frac{1}{2}M). \tag{87}$$

Therefore, the expected construction time is on the order of M^2.

□

3.6 Simulation and Experimental Results

This section introduces two strategies to improve the time efficiency of the BE algorithms. Both the improved BE algorithms and the feedback control law (77) are implemented in MATLAB simulations. We also perform MATLAB simulations of the exploration algorithms and the control law in [18] to compare with the BE algorithms. The BE algorithms and the control law (77) are verified through the experimental results on a Khepera III robot [72].

3.6.1 Simulation Results

Figure 27 depicts the MATLAB simulation results using the exploration algorithms and the control law in [18], and Figure 28 depicts the results using the BE algorithms and the control law (77). In both Figure 27 and Figure 28, the initial position of the vehicle is $(2, 20)$, and the obstacle boundaries are shown in thick red curves. The trajectory of the vehicle is plotted with blue points. Along the vehicle's trajectory, intersections are marked with large green dots.

Figure 27 shows the trajectory of the vehicle using the exploration algorithms and the control law in [18]. In [18] and related works, the vehicle moves through a sector to check whether the sector is open or blocked. In other words, the vehicle moves through a blocked sector until it detects a blocking obstacle boundary. MATLAB simulation results show that 63.4 time units is spent to complete the exploration in Figure 27.

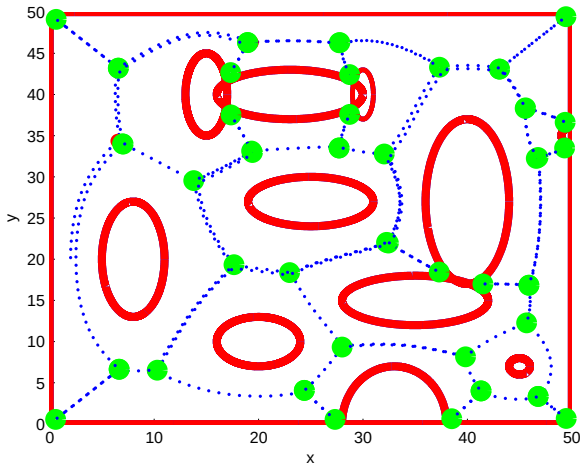


Figure 27: The trajectory of the vehicle built by the exploration algorithms and the control law in [18].

The BE algorithms are theoretically sound, but we can improve the time efficiency of the algorithms without violating the correctness of the algorithms. We have implemented two strategies. The first strategy is inspired by the fact that the vehicle does not have to traverse the entire enclosing boundary to find an unexplored intersection. Whenever the vehicle finishes building a candidate segment, it plans the shortest path to reach the nearest unexplored intersection on the enclosing boundary. Once the vehicle reaches the unexplored intersection, it branches out of the loop to expand the enclosing boundary.

The second strategy is to store the candidate segment with a disabled intersection. If the boundary updating rule is not satisfied, we store the corresponding candidate

segment as a *disabled candidate segment*. Whenever the enclosing boundary is updated, the vehicle checks the disabled candidate segment to see whether there is still an unexplored intersection from the *head* to the *tail*. If no unexplored intersection is found, then the disabled candidate segment will be enabled and boundary expansion can be performed using this candidate segment. This strategy updates the enclosing boundary without letting the vehicle traverse the disabled candidate segment again.

Figure 28 depicts the trajectory of the vehicle using the BE algorithms and the control law (77) with improvement over the time efficiency. The total exploration time is 36.3 time units. The vehicle does not move through a blocked sector, since we assume that the range sensors of the vehicle can detect a blocked sector at an intersection. If this assumption is removed, and we allow the vehicle to detect a blocked sector by retracing behaviors (complete turning whenever the vehicle detects a blocking obstacle boundary) as in [18], then the BE algorithms take 61.8 time units to finish[4]. Hence, for the workspace illustrated in Figure 27 and 28, the time efficiency of the improved BE algorithms is comparable to the algorithms in [18]. Even though more comparison may be necessary to formulate a definite conclusion on comparing the BE algorithms with the algorithms originated from [18], the difference in the behavior of the vehicle is significant enough to justify possible choices made in various contexts. The BE algorithms have added an option to the library of exploration algorithms.

3.6.2 Experimental Results

The validity of our algorithms and the control law (77) is verified using a miniature robot Khepera III [72] that localizes itself based on an odometry system. The Khepera III robot has nine IR sensors and five sonar sensors. In the experiments, we use only IR sensors for range measurements.

[4]We omit the MATLAB figure for this case, since it is almost the same as Figure 27.

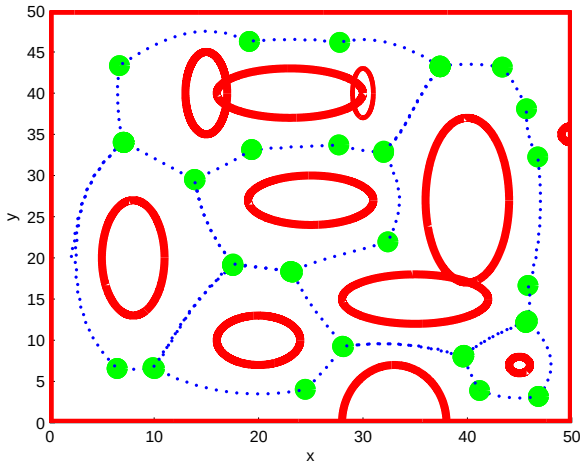


Figure 28: The trajectory of the vehicle built by the BE algorithms and the underlying control law (77).

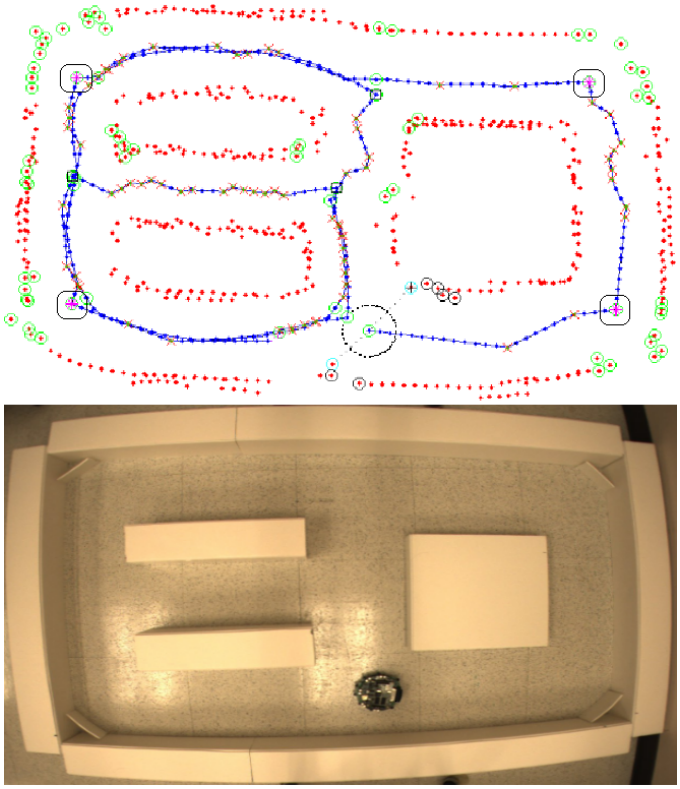


Figure 29: Construction of the Voronoi diagram in a workspace with three rectangular obstacles. The real-time MATLAB plot is displayed above the snapshot of the corresponding obstacle environment. In the MATLAB plot, a rounded rectangle is drawn around each intersection with a blocked sector.

As the robot maneuvers in the workspace, a MATLAB plot is displayed in real time to show the detected obstacle environment. Figure 29 shows the real-time MATLAB plot with the corresponding obstacle environment. The Khepera III robot is depicted as a dotted circle. In addition, the trajectory of the robot is plotted as a blue curve. On the trajectory of the robot, intersections are marked with small circles.

The robot stores coordinates deduced from its odometry and the IR readings for the points detected on the obstacle boundary. These coordinates on the obstacle boundary are referred to as *obstacle points*. To decrease measurement noise in obtaining obstacle points, sensor data are smoothed by convoluting with a Gaussian kernel [64]. The obstacle points derived from IR sensors are shown in red in the MATLAB plot of Figure 29. Despite using Gaussian smoothing to reduce measurement noise, obstacle points are still scattered.

To allow the robot to detect a blocked sector using IR sensors (sensor range:$\sim$ 0.11m), we set up a small piece of cardboard at each corner of the workspace. In the MATLAB plot of Figure 29, green circles centered at obstacle points are used to determine whether a sector is open or blocked. When the robot meets an intersection, green circles appear on the obstacle points that are inside a sector. Hence, by observing the distribution of the green circles, the robot can detect a blocked sector at the intersection. In the MATLAB plot of Figure 29, an intersection with a blocked sector is marked with a magenta star inside a small circle, and a rounded rectangle is drawn around each intersection with a blocked sector. This figure shows that there are intersections with blocked sectors located at the corners of the workspace. Each blocked sector has two end points located on the boundary of the workspace.

When a closest point on either side of the robot is selected by error from the scattered obstacle points, the robot may unexpectedly move off the Voronoi edge and head toward the obstacle boundary using (77). Once the robot is too close to an obstacle on one side, it may not detect an obstacle on the other side due to short

range limitations ($\sim 0.11m$) posted by IR sensors. In this case, instead of using (77), the reactive control [8] is applied for collision avoidance. When the reactive control is applied, the robot's position is marked with "$\times$" in the MATLAB plot (Figure 29).

In the case where the robot has to move along the enclosing boundary that has been constructed previously, the robot follows the enclosing boundary using a method similar to those in [35, 51, 89]. First, we let a virtual vehicle move along the enclosing boundary ahead of the real robot. Then, the real robot keeps moving toward the virtual vehicle to follow the enclosing boundary. Using the virtual vehicle approach, the real robot builds a smoother trajectory than the enclosing boundary initially built. In addition, the real robot can follow the enclosing boundary with higher speed, since the robot does not have to process sensor data while it moves toward the virtual vehicle.

3.7 Conclusions

We develop a provably convergent control law that enables a vehicle to follow Voronoi edges using range sensors. We then develop the boundary expansion algorithms so that the Voronoi diagram structure of an unknown environment can be constructed in finite time. The algorithms implement decisions based on information gathered at each intersection that the vehicle encounters. We prove that such local decisions result in a global behavior that leads to the construction of a complete Voronoi diagram in finite time. Furthermore, we provide an analytic upper bound for the total time spent to construct the Voronoi diagram in a regularized workspace. Simulation and experimental results are provided to demonstrate the effectiveness of both the control law and the exploration algorithms.

CHAPTER IV

SIMULTANEOUS COOPERATIVE EXPLORATION AND NETWORKING BASED ON VORONOI DIAGRAMS

We introduce the SCENT (Simultaneous Cooperative Exploration and NeTworking) algorithms that enable multiple vehicles to cooperatively explore unknown environments. The SCENT algorithms proposed in this chapter construct the Voronoi diagram[1] as a topological map of a workspace. The workspace is considered completely explored if all Voronoi edges are explored. The vehicles are initially deployed at arbitrary locations and are not necessarily aware of the existence of other vehicles. Each vehicle starts with the boundary expansion (BE) algorithms in Chapter 3 to explore its surroundings while deploying information nodes.

Every vehicle deploys information nodes at selected locations while constructing the Voronoi diagram of the explored workspace. As each vehicle builds an information network, the networks built by different vehicles will eventually meet, allowing for inter-vehicle information sharing. This distinguishes the SCENT algorithms from other approaches [10, 44, 39, 11] that only allow robot-to-robot communication.

The SCENT algorithms proposed in this chapter are provably complete under mild technical assumptions. A performance analysis of the SCENT algorithms verifies that in a bounded workspace, the time spent to complete the exploration decreases as the number of vehicles increases. Analytical formulas for this relationship are provided. Furthermore, simulation and experimental results are presented to demonstrate the effectiveness of the SCENT algorithms.

[1] Voronoi diagrams have been widely used for topological maps in robotics, c.f. [18, 73, 76, 15, 61], as well as for studying coverage problems in sensor networks [23, 68].

As a result of the SCENT algorithms, an information network is created concurrently with a topological map of the workspace. The resulting information network and the topological map can then be used as a basis for capturing intruders in the workspace.

In this chapter, we study intruder capturing game on the topological map of the workspace, represented by the Voronoi diagram. We assume that a searcher can access the position of any intruder using the information network. Obeying the conventions established in the literature on graph searching problem [75, 70, 59, 60, 36, 4, 37, 67], an intruder can maneuver at unbounded speed to avoid searchers. Furthermore, an intruder has full knowledge of the environment, positions of the searchers, and the strategies of the searchers. An intruder is *captured* if it is forced to share a node with any searcher.

There are many variants of the graph searching problem originated from [75]. A closely related work to ours is the helicopter cops and robbers game [81, 79]. In this game, it is assumed that the cops have complete knowledge of any robber's position as if the cops are using helicopters. Furthermore, the cops can be placed on or removed from nodes of the graph. A robber is captured when a cop lands on the node occupied by the robber and the robber cannot make any move to escape. A *monotone searching strategy* is a search plan which guarantees that if every robber on one edge is captured, then no robber can enter the edge later. It was found in [79] that if we only consider monotone searching strategies, then the minimum number of cops required depends on the number of robbers.

Similar to a cop using a helicopter in [81, 79], we assume that a searcher detects intruders using the information network. However, we require that a searcher moves along edges of a graph continuously, which is distinct from [81, 79] and is closer to autonomous robot applications. In addition, our searching strategy is not monotone, which implies that even if every intruder on an edge is captured, another intruder may

enter the edge later. Based on this searching strategy, we derive theoretical upper bound for the minimum number of searchers required to capture all intruders on a general graph, which leads to a result on the Voronoi diagram. Note that this upper bound does not depend on the number of intruders. Our searching strategy is further implemented through an interactive online game [49] to assist humans to determine how to secure a complex graph.

This chapter is organized as follows: Section 4.1 introduces preliminaries and background information. Section 4.2 presents the SCENT algorithms and the construction of both the information network and the topological map based on Voronoi diagrams. Section 4.3 discusses the intruder capturing problem utilizing the information network. Section 4.4 demonstrates simulation and experimental results of the SCENT algorithms. Section 4.5 provides conclusions.

4.1 Preliminaries and Background Information

4.1.1 Graph Theory

We review some general notions in graph theory, e.g., [30]. An undirected graph G is defined by a set $G = (N(G), E(G))$, where $N(G)$ denotes the node set and $E(G)$ is a set of unordered pairs of nodes where multiple edges between node pairs are allowed. A *walk* is an alternating sequence of nodes and edges in a graph such that each node belongs to the edge immediately before and after it in the sequence. A graph G is *connected* if there is a walk between every pair of distinct nodes. The *subgraph* of G induced by a set of nodes $S \subset N(G)$ is the pair (S, E_S) where $E_S = \{xy \in E(G) : x, y \in S\}$. A *cycle* is a graph that consists of certain number of nodes connected to form a closed chain.

A graph embedded in the plane without edge crossings is called a *plane graph*. The *faces* of a plane graph are the maximal regions of the plane that contain no point used in the embedding. We say G_D is the *dual graph* of a plane graph G_P if it is constructed

as follows: The nodes of G_D correspond to the faces of G_P. Let $x, y \in N(G_D)$ be two nodes that correspond to the two faces X, Y of G_P. Corresponding to one edge $e \in E(G_P)$ with face X on one side and face Y on the other side, we generated one edge $e^* \in E(G_D)$ connecting two points x and y. The order of the edges in a walk around the boundary of X of G_P is the order of the corresponding edges incident to $x \in N(G_D)$.

An *edge cover* of G, $E_C(G) \subset E(G)$, is a set of edges such that every node in G is incident to some edge in $E_C(G)$. We say that all nodes in G are *covered* by the edges in $E_C(G)$ if $E_C(G)$ is an edge cover. The notion $\alpha(G)$ denotes an edge cover of G with the minimum cardinality, i.e., the fewest number of edges.

4.1.2 The Workspace and Its Voronoi Diagram

Consider a connected and compact workspace $W \subset R^2$ whose boundary, ∂W, is a simple closed curve. In other words, ∂W is continuous and no self-intersection occurs. Let O_1,O_2,...O_{M-1} be $M-1$ disjoint, and compact obstacles such that $O_i \subset W$. We introduce O_M as a "virtual" obstacle that bounds the workspace, i.e., $\partial W \subset \partial O_M$. We denote the set of obstacles as $S_O = \{O_1, O_2, ...O_M\}$.

Obeying the conventions established in the literature on Voronoi diagrams [2, 73, 61, 58, 14, 5], we define the *Voronoi cell* for an obstacle O_i as the set of points that are closer to O_i than to any other obstacle in S_O for $i \in \{1, 2, ..., M\}$. $\partial V(O_i)$ is the boundary of the Voronoi cell for O_i, i.e., $V(O_i)$.

The following assumptions, which are also used in Chapter 3, are made about the workspace: $\bigcup_{O_i \in S_O} \overline{V}(O_i) = W$ where $\overline{V}(O_i) = V(O_i) \bigcup \partial V(O_i)$, and $\partial V(O_i)$ is a simple closed curve for each $O_i \in S_O$, i.e., $\partial V(O_i)$ is continuous and no self-intersection occurs.

The *Voronoi diagram* $V = (N(V), E(V))$ is defined as the union of all cell boundaries. Since $\partial V(O_i)$ is a simple closed curve for each $O_i \in S_O$, Voronoi cells correspond

to faces in V. Let a *cycle basis* denote a cycle in V enclosing a single Voronoi cell. A *Voronoi edge* in $E(V)$ is a common boundary edge shared by two Voronoi cells $V(O_i)$ and $V(O_j)$. A *Voronoi vertex* in $N(V)$ is a point where more than two Voronoi edges meet.

We define an *intersection* P as a point at which the following condition is satisfied:

- there exists a circle centered at P intersecting obstacle boundaries at more than two points. The interior of the circle does not intersect any obstacles.

Since an intersection is equidistant from more than two points on obstacle boundaries, any Voronoi vertex is an intersection.

Suppose there is a cycle C contained in the Voronoi diagram V. $V[C]$ is the subgraph of V enclosed by C. Let $V^*[C]$ denote the dual graph of $V[C]$ with the node corresponding to the unbounded face removed. We define $n(O_i)$, where O_i is an obstacle enclosed by C, as the node in $V^*[C]$ corresponding to the Voronoi cell $V(O_i)$.

Let V^* be $V^*[C]$ when we choose $\partial V(O_M)$ as C, i.e., V^* is the dual graph of V with the node representing the unbounded face removed. Let $\partial(V^*[C])$ denote the subgraph of $V^*[C]$ induced by the set of nodes on the boundary of the unbounded face of $V^*[C]$. The relationship among the subgraphs is that $\partial(V^*[C]) \subset V^*[C] \subset V^*$. See Figure 30 for the illustration of $V[C]$, $V^*[C]$, and $\partial(V^*[C])$. Lemma 2 states that $V^*[C]$ is connected.

Lemma 2. *Let C be a cycle in the Voronoi diagram V. Then, $V^*[C]$ is connected.*

Proof. By contradiction, we prove that if $V^*[C]$ is not connected, then C cannot be a simple closed curve. Suppose a node in $H_x \subset N(V^*[C])$ is not connected to a node in $H_y = N(V^*[C]) \setminus H_x$. This further implies that no edge is shared by $\bigcup_x V(O_x)$ and $\bigcup_y V(O_y)$, where $V(O_x)$ and $V(O_y)$ are Voronoi cells corresponding to nodes in H_x and H_y respectively. Since every node in $V^*[C]$ has corresponding Voronoi cell

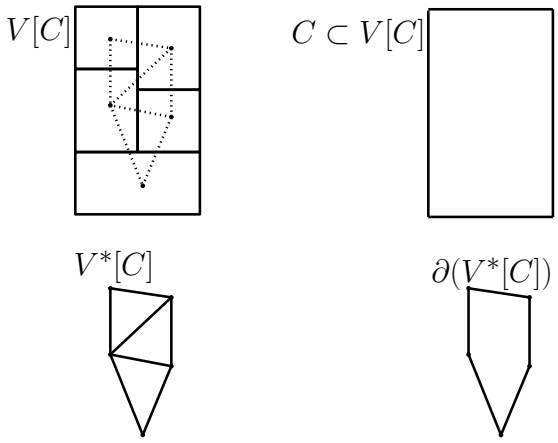


Figure 30: C, $V[C]$, $V^*[C]$, and $\partial(V^*[C])$.

enclosed by C, C must enclose both $\bigcup_x V(O_x)$ and $\bigcup_y V(O_y)$. However, C cannot be a simple closed curve, since $\bigcup_x V(O_x)$ and $\bigcup_y V(O_y)$ share no common edge. □

4.1.3 The BE Algorithms

In Chapter 3, we introduce the BE algorithms to construct the Voronoi diagram of the environment with one vehicle equipped with range sensors. We review the BE algorithms briefly. Suppose the vehicle can track Voronoi edges using the tracking control law in Chapter 3. The vehicle can circle around an obstacle, say O_i, to its right while traversing the boundary of $V(O_i)$. We denote the cycle enclosing O_i as the *initial enclosing boundary* B_0. Then, we expand the enclosing boundary by adding one obstacle at each step of the algorithm, until the boundary encloses all obstacles except for O_M. We use B_k to denote the enclosing boundary obtained after k steps. B_k has $k+1$ obstacles inside it and is maintained as a simple closed curve tracked by the vehicle in the clockwise direction. Let an *unexplored intersection* denote an intersection incident to an unvisited Voronoi edge. The BE algorithms guarantee that there is no unexplored intersection strictly inside the boundary. It is proved that the BE algorithms end in finite time resulting in a complete Voronoi diagram.

4.2 The SCENT Algorithms

In this section, we present the SCENT algorithms by extending the BE algorithms for one vehicle to multiple vehicles. Let us denote each vehicle as v^i where $1 \leq i \leq N_v$ and N_v is the number of vehicles. Let B^i denote the enclosing boundary built by v^i. The *enclosure of* B^i, $cl(B^i)$, denotes the region inside B^i. Since multiple vehicles are involved, we modify the BE algorithms to expand B^i in such a way that $cl(B^i)$ does not overlap with $cl(B^j)$ for any $j \neq i$. When v^i visits an intersection on $B^i \bigcap B^j$, v^i moves along the edges $B^i \bigcap B^j$ but does not move into $cl(B^j)$. This in turn avoids overlaps.

In the SCENT algorithms, every vehicle v^i deploys information nodes on intersections. If necessary, information nodes are deployed on long Voronoi edges in order to relay data from one node to another node that is out of the maximum communication range. These information nodes then form an information network.

We assume that nodes can be deployed at desired locations and that the information network is in place once the nodes are deployed. Here, each information node has unique ID so that a vehicle visiting a node can localize itself using the unique ID of the node. We further assume that each information node stores an unexplored edge emanating from the node. This is a feasible assumption, since each node can detect Voronoi edges emanating from the node based on sensor footprint information.

4.2.1 Information Graph

We define an *information graph*, I, as the graph where every node represents a deployed information node and every edge represents a communication link between nodes. A new node and edges are added to the graph when a vehicle deploys an information node. For each vehicle v^i, we define $I^i \subset I$ as the information graph being constructed by v^i. In addition, we define $I(B^i)$ as the subgraph of I^i, whose node set corresponds to the set of information nodes deployed along B^i. In other words, $I(B^i)$

is the communication infrastructure along B^i. Suppose that an information node is already deployed by v^i at an intersection P and that v^j visits P later. Then, through the information node deployed at P, v^i can relay the data structure $I(B^i)$ to v^j and vice versa. Note that v^j does not have to drop an information node at P, since a communication link is already established between the node at P and the network built by v^j.

Once a communication link is established between a node deployed by v^i and a node deployed by v^j, v^i ($i \in \{1, 2, ..., N_v\}$) builds the maximally connected graph $I^i = (N(I^i), E(I^i)) \subset I$ such that $I^i = I^j$. Every vehicle v^i stores I^i, $I(B^i)$, and $I(B^j)$ where j is determined such that $I(B^j) \subset I^i$.

4.2.2 Avoid Blocking Using the Coverage Graph

Even though v^i expands B^i in such a way that $cl(B^i)$ does not overlap with $cl(B^j)$ for any $j \neq i$, there may be the case where the expanding enclosing boundary B^i is blocked by the enclosing boundaries constructed by other vehicles. *Blocking* of B^i denotes the situation when $B^i \subset \bigcup_{j \neq i} B^j \bigcup \partial V(O_M)$.

To avoid a blocking situation, each vehicle v^i ($i \in \{1, 2, ..., N_v\}$) builds a *coverage graph* Γ^i from the information graph. The union of graphs $\bigcup_{I(B^j) \subset I^i} I(B^j)$ can be embedded in R^2 without crossing, since we expand the enclosing boundary for v^i in such a way that the enclosure built by v^i does not overlap with the enclosures built by other vehicles. Let G^i be the dual graph of embedded $\bigcup_{I(B^j) \subset I^i} I(B^j)$. Since $cl(B^i)$ denotes the region inside B^i, an enclosure corresponds to one node in G^i. Γ^i denotes the subgraph of G^i induced by the nodes corresponding to enclosures. Let P_j denote the node in Γ^i where j is determined to satisfy that $I(B^j) \subset I^i$. Since a subgraph of a plane graph is a plane graph, Γ^i is a plane graph. Multiple edges arise in Γ^i when distinct enclosures have more than one common boundary edge. See Figure 31 for the illustration of Γ^i.

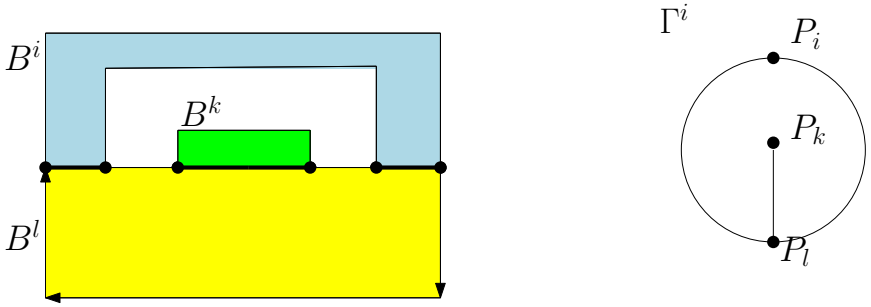


Figure 31: Construction of the coverage graph from enclosing boundaries. Left: enclosing boundaries built by distinct vehicles. A common boundary edge between two distinct enclosures is marked with a bold edge. The expansion of B^k will be eventually blocked by the enclosing boundaries, B^l and B^i, constructed by other vehicles. Right: the coverage graph, where P_k is surrounded by a cycle formed by edges connecting P_i and P_l.

The following theorem shows that a cycle surrounding a node P_k indicates the blocking of B^k.

Theorem 6. *Suppose that a node P_k is surrounded by a cycle contained in Γ^i. Each vehicle expands its enclosure using the BE algorithms in Chapter 3 while satisfying that its enclosure does not overlap with the enclosures built by other vehicles. Then, the blocking of B^k will eventually occur.*

Proof. Suppose that a node P_k is surrounded by a cycle, say C, contained in Γ^i. Let $N(C)$ denote the node set of C. Suppose that P_k is surrounded by C and that $N(C) = \{P_1, P_2, ...P_{|N(C)|}\}$ where $k > |N(C)|$ to indicate that P_k is not in $N(C)$. In a walk along C, $P_1, P_2, ...P_{|N(C)|}$ are encountered in this order.

According to the definition of a dual graph, there exists a common boundary edge between $cl(B^j)$ and $cl(B^{j+1})$ where $j \leq |N(C)|$. Thus, the area surrounded by $\overline{cl}(B^1) \bigcup \overline{cl}(B^2) ... \bigcup \overline{cl}(B^{|N(C)|})$ is finite. Since $\Gamma^i \subset G^i$, P_k surrounded by $C \subset \Gamma^i$ indicates that $cl(B^k)$ exists within the finite area surrounded by $\overline{cl}(B^1) \bigcup \overline{cl}(B^2) ... \bigcup \overline{cl}(B^{|N(C)|})$.

Using the BE algorithms, each enclosure only expands but does not shrink. Thus, the area surrounded by $\overline{cl}(B^1) \bigcup \overline{cl}(B^2) ... \bigcup \overline{cl}(B^{|N(C)|})$ does not increase as time goes on. Since each vehicle expands its enclosure while satisfying that its enclosure does

not overlap with the enclosures built by other vehicles, the expansion of $cl(B^k)$ will be blocked by the set $\overline{cl}(B^1) \bigcup \overline{cl}(B^2) ... \bigcup \overline{cl}(B^{|N(C)|})$. □

For $P_l \in N(\Gamma^i)$, clockwise cyclic ordering of its neighboring nodes is determined by sweeping along B^l in the clockwise direction. For example, in Figure 31, sweeping along B^l in the clockwise direction gives common boundary edges $B^l \bigcap B^i \rightarrow B^l \bigcap B^k \rightarrow B^l \bigcap B^i$ which provides clockwise cyclic ordering for P_l as $P_i \rightarrow P_k \rightarrow P_i$. Since cyclic ordering of neighbors can be determined for every node in Γ^i, we can detect a cycle surrounding a node P_k with the SweepCycle algorithm proposed in [42].

If a node P_k is surrounded by a cycle, then the blocking of B^k will eventually occur according to Theorem 6. To avoid a blocking situation, we propose to let v^i obey the *cycle avoiding rule* as follows.

- If expansion of B^i_n, which denotes the enclosing boundary for v^i updated after n steps, leads to P_k being surrounded by a cycle for some k, then the expansion will not be performed.

The left sub-figure of Figure 32 illustrates the case where we avoid a cycle that surrounds P_k on Γ^i. Even if v^i builds a candidate segment using the BE algorithms depicted as lines with arrows, the expansion of B^i is not performed to avoid a cycle that surrounds P_k on Γ^i. This prevents the situation in Figure 31.

4.2.3 Resolve Blocking or Overlapping Situations

The procedure in detecting cycles only works for Γ^i built from I^i. There exist situations where I^l is not connected to I^i as illustrated in the right sub-figure of Figure 32. In this case, the existence of I^l is unknown to the vehicles that are only aware of I^i. Hence, the expansion of B^l will eventually be blocked.

Another situation is *overlapping*. This can happen if the initial enclosing boundaries built by two different vehicles are identical. This is possible at the beginning of

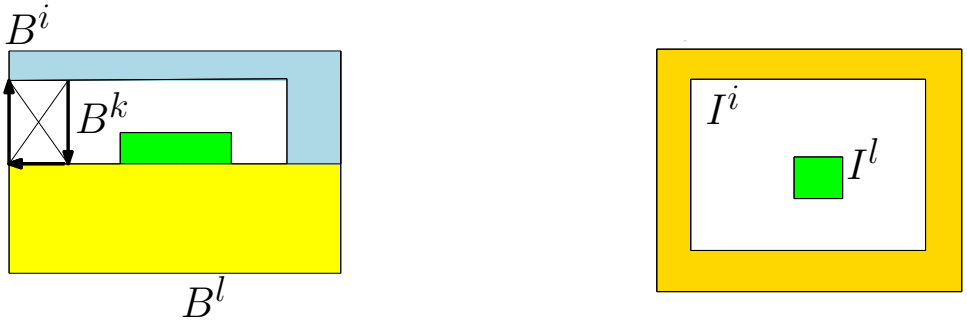


Figure 32: The left sub-figure shows the case where we avoid the blocking of B^k using the cycle avoiding rule. The right sub-figure shows the case where I^l is not connected to I^i.

the exploration, if the initial position of v^i is such that the obstacle to the right of v^i is also to the right of v^j.

In the case where blocking or overlapping is detected, the blocked or overlapped vehicle can be redirected to an unexplored intersection where a new enclosing boundary can be built. An unexplored intersection corresponds to a node in I^i storing an unexplored edge emanating from the node. As long as there exists an unvisited Voronoi edge in W, every vehicle v^i can find a node in I^i storing an unexplored edge emanating from the node. This is stated in Lemma 3.

Lemma 3. *If there exists an unvisited Voronoi edge in W, then every vehicle v^i can find a node in I^i storing an unexplored edge emanating from the node.*

Proof. We prove by contradiction. Suppose that every node in I^i stores no unexplored edge emanating from the node. Thus, the edge set of I^i does not contain any edges that lead to unexplored regions. This can only be true if I^i has all Voronoi edges in W, which implies that all Voronoi edges in W have been visited by vehicles. This is a contradiction. □

The redirecting strategy works as follows. When v^i detects blocking or overlapping, then v^i searches for an unexplored intersection on I^i. Note that this unexplored intersection will not lie on a blocked enclosing boundary. This is stated as Lemma 4.

Lemma 4. *If an unexplored intersection is found on I^i, then this unexplored intersection is not on a blocked enclosing boundary.*

Proof. We prove by contradiction. Suppose that an unexplored intersection is found on a blocked enclosing boundary B^i. This implies that there exists an unvisited edge intersecting the unexplored intersection. This unvisited edge leads outside of B^i, because no unexplored intersection is strictly inside B^i (Theorem 4 in Section 3.4). Since B^i is blocked, we have $B^i \subset \bigcup_{n \neq i} B^n \bigcup \partial V(O_M)$. Hence, the unvisited edge must lead inside B^n where $B^i \bigcap B^n \neq \emptyset$. However, this unvisited edge cannot lead inside B^n either, because no unexplored intersection is strictly inside B^n (Theorem 4 in Section 3.4). Therefore, an unexplored intersection cannot exist on a blocked enclosing boundary. □

By applying the breadth-first search algorithm on I^i, v^i can find the shortest (hop distance) path from the current position of v^i to every unexplored intersection on I^i. Among these unexplored intersections, v^i selects the one with the smallest hop distance and marks it as r_{v^i}. The position of r_{v^i} is relayed (broadcasted) across I^i to all other vehicles sharing I^i. In the case where v^j (for any $j \neq i$) visits r_{v^i}, v^j ignores r_{v^i} without changing B^j. In this way, r_{v^i} is "reserved" for v^i until it is reached by v^i. Note that v^i moves along the shortest path to reach r_{v^i}. Once v^i reaches r_{v^i}, it starts building a new enclosing boundary.

This strategy relies on the availability of at least one unexplored intersection which has not been reserved by any other vehicle. Hence, we make the following assumption:

(S1) When blocking or overlapping occurs for v^i, there exists at least one unexplored intersection on I^i, which has not been marked as r_{v^j} by some other vehicle v^j (for any $j \neq i$).

The SCENT algorithms are described in algorithm 3 and algorithm 4. Data structures and operations used in the SCENT algorithms are summarized in Table 2.

Table 2: Data Structures and Operations

L_u: the singly linked list representing the enclosing boundary under construction.
L_u.Insert(P): insert P at the end of the linked list L_u.
L: the circularly linked list representing the current enclosing boundary for a vehicle v^i.
HT=L.seg(*head*,*tail*): the segment of L that starts from the *head* and ends at the *tail*.
L_r= L.Remove(HT): remove the linked list HT from L resulting in L_r.
CS: the singly linked list representing the candidate segment.
L=L_r.Combine(CS): combine the linked list L_r with CS resulting in the updated enclosing boundary L.
HT.Search(*unexplored*): search for unexplored intersections in the linked list HT. If there is no unexplored intersection, return $NULL$.
CS.Update: if the boundary updating rule and the cycle avoiding rule are satisfied, then return $TRUE$. Otherwise, return $FALSE$.
D_k^i: the disabled intersection set of B_k^i which denotes the enclosing boundary for v^i updated after k steps.
Set.Store(*Data*): store *Data* in *Set*.
R^i: the set of reserved intersections (r_{v^j} where $I^j == I^i$) stored in a vehicle v^i.
v^i.Move(r_{v^i}): v^i chooses r_{v^i} among unexplored intersections on I^i. r_{v^i} is relayed to every vehicle sharing I^i. v^i moves along the shortest path for reaching r_{v^i} followed by moving through an unvisited edge at r_{v^i}.

Head, *tail*, open sector, sector 0, sector selection rule, disabled intersection, candidate segment(CS), and boundary updating rule are discussed in Chapter 3.

4.2.4 Performance Analysis

We provide an analytical upper bound for the total time spent to construct the Voronoi diagram using the SCENT algorithms in a regularized workspace. As the number of Voronoi cells in a bounded workspace increases, each Voronoi cell approaches to hexagonal shape [31, 74]. Thus, we analyze the performance of our algorithms in the workspace where each cell has a hexagonal shape with identical size. In the appendix, we state the conditions for a workspace to obtain hexagonal Voronoi cells.

The SCENT algorithms for v^i end when no unexplored intersection is detected using I^i or the total number of obstacles inside enclosing boundaries built by v^i is

Algorithm 3 Construct the Initial Enclosing Boundary for v^i

```
n ← 1;
repeat
  v^i encounters an intersection;
  P_{n,0} ← the intersection;
  search for an open sector in the counterclockwise direction from sector 0;
  v^i moves through the first open sector;
  P_{n,0}.pointer← first open sector;
  if P_{n,0} has an open sector that has not been visited by vehicles then
    P_{n,0}.mark← unexplored;
  else
    P_{n,0}.mark← explored;
  end if
  L_u.Insert(P_{n,0}); n ← n + 1;
until v^i encounters the P_{1,0} for the second time;
L ← L_u;
if L == B^m for any m ≠ i then
  v^i.Move(r_{v^i}); repeat this algorithm;
else
  implement algorithm 4;
end if
```

bigger than $\lceil \frac{M-1}{N_v} \rceil$. Here, the ceiling function is used since the number of obstacles is a positive integer.

Theorem 7 discusses the upper bound for the exploration time using the SCENT algorithms. Before presenting Theorem 7, we introduce Lemma 5, which is based on Theorem 5 in Section 3.5. Recall that assumptions (A1)-(A4) are presented in Section 3.3.1.

Lemma 5. *Consider unit speed vehicles and workspace W with assumptions (A1)-(A4) satisfied. All obstacles, except for O_M, have hexagonal Voronoi cells with identical size. Suppose that every vehicle explores W using the SCENT algorithms. In addition, suppose a blocking has not occurred until the enclosing boundary is updated with K steps. Then, the time to update the enclosing boundary with K steps is bounded above by $T(\frac{1}{2}K^2 + \frac{3}{2}K + 1)$ where T denotes the time for a vehicle to traverse along the edges of one hexagonal Voronoi cell.*

Algorithm 4 Expand the Enclosing Boundary for v^i

$n \leftarrow 1$; $k \leftarrow 0$;
while there exists an unexplored intersection on I^i **do**
  v^i visits $P_{n,k}$ on L;
  if $P_{n,k} \notin R^i \bigcup D_k^i$ and there exists an open sector, which does not lead into B^j for any $j \neq i$, satisfying the sector selection rule R1 in Chapter 3 **then**
    $m \leftarrow 1$; $S_1 \leftarrow P_{n,k}$; $MeetTail \leftarrow 0$;
    while $MeetTail \neq 1$ **do**
      v^i finds S_m;
      if $m == 1$ **then**
        move through the sector chosen using rule R1 in Chapter 3;
      else
        move through the sector chosen using rule R2 in Chapter 3;
      end if
      S_m.pointer$\leftarrow$ the selected sector;
      if S_m has an open sector that has not been visited by vehicles **then**
        S_m.mark$\leftarrow$ *unexplored*;
      else
        S_m.mark$\leftarrow$ *explored*;
      end if
      CS.Insert(S_m);
      if $m \neq 1$ and $S_m == P_{T,k}$ for any T **then**
        $MeetTail \leftarrow 1$;
      else
        $m \leftarrow m + 1$;
      end if
    end while
    $head \leftarrow S_1$; $tail \leftarrow P_{T,k}$; HT=L.seg($head, tail$);
    if CS.Update== $TRUE$ **then**
      L_r=L.Remove(HT); L=L_r.Combine(CS);
      $P_{1,k+1} \leftarrow tail$; N is the number of intersections on L; relabel the intersections on L in the clockwise direction as $P_{1,k+1}$,$P_{2,k+1}$,...,$P_{N,k+1}$;
      $n \leftarrow 1$; $k \leftarrow k + 1$;
    else
      D_k^i.Store($head$); $n \leftarrow T$;
    end if
  else
    $n \leftarrow n + 1$;
  end if
  if $(L \subset \bigcup_{q \neq i} B^q \bigcup \partial V(O_M))$ **then**
    v^i.Move(r_{v^i}); repeat algorithm 3;
  end if
  if v^i receives r_{v^l} from another vehicle v^l sharing I^i ($I^i == I^l$) **then**
    R^i.Store(r_{v^l});
  end if
end while

Theorem 7. *We consider N_v unit speed vehicles and workspace W with assumptions (A1)-(A4) and (S1) satisfied. Suppose that there are M obstacles such that all obstacles, except for O_M, have hexagonal Voronoi cells with identical size. Also, suppose that every vehicle explores W using the SCENT algorithms. Let T denote the time for a vehicle to traverse along the edges of one hexagonal Voronoi cell. Let $T_0(M)$, where M is used to indicate that T_0 is a function of M, denote the time required for a vehicle to traverse along the perimeter of W. Then, the time to construct a complete Voronoi diagram is bounded above by $T + \lceil \frac{M-1}{N_v} \rceil (T_0(M) + T) + \frac{3}{2} T \lceil \frac{M-1}{N_v} \rceil + \frac{1}{2} T (\lceil \frac{M-1}{N_v} \rceil)^2$.*

Proof. Suppose every vehicle v^i reaches the moment when the total number of obstacles inside enclosing boundaries built by v^i is bigger than $\lceil \frac{M-1}{N_v} \rceil$. Since N_v vehicles are deployed in W and $M-1$ obstacles are surrounded by $\partial V(O_M)$, a complete Voronoi diagram in W is built accordingly. Therefore, in order to obtain an upper bound for the time to build a complete Voronoi diagram, we will derive a time upper bound for one vehicle v^i to reach the moment when the total number of obstacles inside enclosing boundaries built by v^i is bigger than $\lceil \frac{M-1}{N_v} \rceil$.

Suppose that blockings occur l times before the SCENT algorithms for v^i terminate. Since the number of obstacles is finite, l is also finite. Let K_j denote the updated step of the enclosing boundary between $j-1$th blocking and jth blocking. In other words, after the enclosing boundary is updated with K_j steps, a blocking occurs and a new enclosing boundary is built at a new position. Then, the enclosing boundary is updated with K_{j+1} steps before a blocking occurs again. In this way, boundary updates occur in the order of $K_1 \rightarrow ... \rightarrow K_{l+1}$ and blockings occur l times in total.

Figure 33 illustrates the case where blockings for v^i occur two times in a rectangular shaped workspace W. Boundary updates occur in the order of $K_1 \rightarrow K_2 \rightarrow K_3$. The region inside B_{K_i} is shaded in Figure 33. When a blocking occurs, v^i is redirected to r_{v^i} and builds a new enclosing boundary. Redirection of v^i is depicted as an arrow

in this figure.

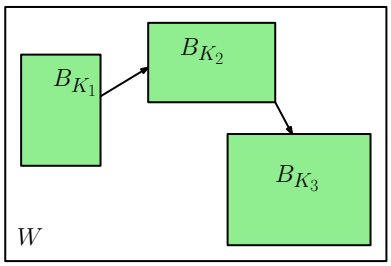


Figure 33: The case where blockings for v^i occur two times.

Let T_{K_j} denote the time to update the enclosing boundary for v^i with K_j steps before a blocking occurs. Using Lemma 5, we derive an upper bound for T_{K_j} as follows.

$$T_{K_j} \quad \leq T(\tfrac{1}{2}K_j^2 + \tfrac{3}{2}K_j + 1). \tag{88}$$

Suppose a blocking occurs after the enclosing boundary is updated with K_j steps. When a blocking occurs, v^i is redirected to the nearest unexplored intersection r_{v^i} and builds a new enclosing boundary. Since v^i moves along the shortest path to reach r_{v^i}, the time to reach r_{v^i} is upper bounded by $T_0(M)$, which is the time required for a vehicle to traverse along the perimeter of W.

Since blockings occur l times in total, we get an upper bound of exploration time as follows.

$$T_c \quad \leq T_{B_0} + T_0(M) + \textstyle\sum_{j=1}^{l+1} T(\frac{1}{2}K_j^2 + \frac{3}{2}K_j + 1) + lT_0(M), \tag{89}$$

where (88) is used as the time for a vehicle to update the enclosing boundary with K_j steps. On RHS of (89), $T_{B_0} + T_0(M)$ is added considering the overlapping of B^i. Here, T_{B_0} is the time required to build the initial enclosing boundary, since an overlapping of B^i is detected as the initial enclosing boundary is built. Once an overlapping of B^i is detected, v^i chooses r_{v^i} and moves along the shortest path to reach r_{v^i}.

Next, we derive an equation for $\sum_{j=1}^{l+1} K_j$ in (89) using N_v, l, and M. The SCENT algorithms for v^i end when the total number of obstacles inside enclosing boundaries

built by v^i is $\lceil \frac{M-1}{N_v} \rceil$. Using the fact that there are $K+1$ obstacles inside B_K, the SCENT algorithms for v^i finish when

$$\sum_{j=1}^{l+1}(K_j+1) = \lceil \frac{M-1}{N_v} \rceil. \tag{90}$$

Rearranging LHS of (90), we obtain

$$\sum_{j=1}^{l+1} K_j = \lceil \frac{M-1}{N_v} \rceil - l - 1 \geq 0, \tag{91}$$

where inequality holds, since $K_j \geq 0$ for all j.

Using (91), (89), and the fact that $\sum_{j=1}^{l+1}(K_j^2) \leq (\sum_{j=1}^{l+1} K_j)^2$, we obtain

$$T_c \ < T_{B_0} + (l+1)T_0(M) + (l+1)T + \tfrac{3}{2}T(\lceil \tfrac{M-1}{N_v} \rceil - l - 1) + \tfrac{1}{2}T(\lceil \tfrac{M-1}{N_v} \rceil - l - 1)^2. \tag{92}$$

Furthermore, using (91) and $T_{B_0} = T$, we get

$$T_c \ < T + \lceil \tfrac{M-1}{N_v} \rceil (T_0(M) + T) + \tfrac{3}{2}T\lceil \tfrac{M-1}{N_v} \rceil + \tfrac{1}{2}T(\lceil \tfrac{M-1}{N_v} \rceil)^2. \tag{93}$$

□

In the case where $\frac{M-1}{N_v} \gg 1$, (93) is approximated as

$$T_c < \tfrac{M-1}{N_v}(T_0(M)) + \tfrac{1}{2}T(\tfrac{M-1}{N_v})^2 \ . \tag{94}$$

Suppose that $\frac{M-1}{N_v}(T_0(M))$ is a dominant term on RHS of (94). In this case, the upper bound of exploration time decreases by a factor of m as N_v increases by a factor of m. However, in the case where $\frac{1}{2}T(\frac{M-1}{N_v})^2$ is a dominant term on RHS of (94), the upper bound of exploration time decreases by a factor of m^2 as N_v increases by a factor of m.

Next, consider the case where N_v increases beyond $M-1$ ($\lceil \frac{M-1}{N_v} \rceil \to 1$). Then, from (93), we obtain

$$T_c < T_0(M) + 4T, \tag{95}$$

which implies that, as the number of vehicles increases beyond $M-1$, the effectiveness of adding more vehicles decreases. We also acknowledge that, as N_v increases, assumption (S1) gets more difficult to be satisfied.

4.3 Capturing Intruders Using the Information Network

As a result of the SCENT algorithms, an information network is created concurrently with a topological map based on Voronoi diagrams. We utilize the constructed network as a basis for capturing intruders in the workspace. We consider a simplified scenario such that searchers and intruders move along edges of the Voronoi diagram.

4.3.1 Definitions and Assumptions

Let us consider an environment where searchers and intruders move along edges of a graph G. Suppose that only one searcher, called the *free searcher*, moves along edges of a graph continuously. The free searcher obtains the position of any intruder whenever the searcher visits a node in a graph. This is feasible, since the free searcher utilizes the information network to detect intruders. We further introduce a *guard*, whose role is to guard a node in a graph.

Obeying the conventions established in the literature on capturing intruders on graphs [70, 55, 56, 6, 27], we assume that an intruder has full knowledge of the environment, searching strategies, and positions of both the free searcher and guards. In addition, an intruder moves along edges of G at unbounded speed to avoid both the free searcher and guards.

An intruder is *captured* if (1) it is on an edge whose one end is guarded while the free searcher moves through the edge from the opposite end, or (2) it is on an edge one end of which has degree one while the free searcher moves through the edge starting from the opposite end. Denote the minimum number of guards to capture all intruders on G as $g_I(G)$ where I indicates the information network.

4.3.2 Capturing Intruders on a General Graph

In this subsection, we derive an upper bound for $g_I(G)$ for a general graph G. The authors of [79] introduced a searching strategy to capture one intruder on a tree graph T using one free searcher. Suppose n is a node of T. Then, we define a *branch* of T at n as the maximal subtree of T, denoted by T', if n has degree one in T'. Lemma 6 provides the searching strategy [79].

Lemma 6. *Suppose that one free searcher and one intruder move along a tree graph T. Then, the searcher can capture the intruder in finite time using the following strategy:*

Whenever the searcher meets a node, it obtains the position of the intruder. Then, it chooses the branch containing the intruder and moves through the edge contained in the branch until it meets another node. Iterate this until the intruder is captured.

Since an intruder can move at unbounded speed, an intruder can escape from the free searcher using a cycle in a general graph G. To block the escape of an intruder, we deploy guards at nodes in G. If a guard is deployed at a node, then the node becomes unavailable to intruders. Hence, once a guard is deployed at a node, we mark the node as *guarded*. We say that a cycle C is *blocked* if any node in $N(C)$ is guarded.

We define MinGuard(G) as the minimum number of guarded nodes to block all cycles contained in G. MinGuard(T)=0 for a tree graph T. For a general graph G, computational method to search for MinGuard(G) corresponds to a set cover optimization problem which is known to be NP-hard in computer science. For this set cover optimization problem, approximation algorithms returning near-optimal solutions exist [33]. In our problem of searching for MinGuard(G), the universal set U corresponds to the set of all cycles that are contained in G. Let a node in G be identified by n_i where $i \leq |N(G)|$. We build a family of subsets $S = \{S_1, S_2, ..., S_{|N(G)|}\}$

of U. Here, S_i is the set of cycles satisfying that every cycle in S_i has n_i in it. Hence, every cycle in S_i is blocked by deploying a guard at $n_i \in N(G)$. $P \subset S$ is selected so that it contains all cycles in U. Then, MinGuard(G) corresponds to the minimum number of the sets in P.

Suppose MinGuard(G) nodes of G are guarded to block all cycles contained in G. Since all cycles in G are blocked, no cycle is available to an intruder, i.e., available set of points for an intruder is a tree. Thus, an intruder cannot escape from the free searcher using Lemma 6. Furthermore, the free searcher can capture all intruders by chasing one intruder at a time. Since we need one free searcher and MinGuard(G) guards to capture all intruders on G, $g_I(G) \leq$MinGuard(G).

Lemma 7. *For a general graph G, $g_I(G) \leq$MinGuard(G).*

For a general graph G, algorithm 5 describes the graph searching strategy using one free searcher and MinGuard(G) guards. Algorithm 5 is also implemented through an interactive online game [49] to assist humans to determine how to secure a complex graph. The game uses a greedy (approximation) algorithm [33] to select which nodes should be guarded so as to block all cycles in a given graph.

Algorithm 5 Capturing All Intruders on a General Graph G

```
guard MinGuard(G) nodes of G to block all cycles in G;
s_f ← one free searcher which maneuvers to capture intruders;
repeat
  s_f chooses one intruder as TARGET;
  T_D ⊂ G ← the tree which is available to TARGET;
  if s_f is not on T_D then
    s_f moves along G to reach a node in T_D;
  end if
  repeat
    s_f meets a node in T_D and obtains the position of TARGET;
    s_f chooses the branch of T_D containing TARGET and moves through the edge
    contained in the branch until it meets another node;
  until s_f captures TARGET, and TARGET is removed from the list of intruders;
until all intruders on G are captured;
```

4.3.3 Capturing Intruders on Voronoi Diagrams

In this subsection, we make use of the specified Voronoi diagram $V = (N(V), E(V))$ as introduced in Section 4.1.2 to study $g_I(V)$.

Recall that V^* is the dual graph of V with the node representing the unbounded face removed. In Lemma 2, $V^* = V^*[C]$ if we choose $\partial V(O_M)$ as C. According to Lemma 2, V^* is a connected graph.

Depending on the structure of V, V^* can be one node or a connected graph with more than one node. In the case where V^* is a connected graph with more than one node, an edge cover of V^* exists. Hence, we can derive an upper bound for $g_I(V)$ using an edge cover of V^*. However, in the case where V^* is one node, an edge cover of V^* cannot exist.

In the following lemma, we obtain $g_I(V)$ in the case where V^* is one node.

Lemma 8. *For the Voronoi diagram $V = (N(V), E(V))$ such that V^* is one node, $g_I(V) = 1$.*

Proof. Since V^* is one node, there is only one cycle in V. Since an intruder can move along this cycle at unbounded speed to avoid the free searcher, we require one guard on the cycle, i.e., $g_I(V) = 1$. □

Before tackling the case where V^* is a connected graph with more than one node, we consider the special case where V^* is a tree graph with more than one node. In this special case, we use algorithm 6 to guard $|\alpha(V^*)|$ nodes in V. Recall that $\alpha(G)$ denotes an edge cover of G with the minimum cardinality. In the following theorem, we prove that guarding $|\alpha(V^*)|$ nodes using algorithm 6 blocks all cycles in V.

Theorem 8. *For the Voronoi diagram $V = (N(V), E(V))$ such that V^* is a tree graph with more than one node, $g_I(V) \leq |\alpha(V^*)|$.*

Algorithm 6 Guarding $|\alpha(V^*)|$ nodes in V (In the case where V^* is a connected graph with more than one node, we replace $\alpha(V^*)$ in this algorithm by $\beta(V^*)$)

the edge set $E_C \leftarrow \alpha(V^*)$;
repeat
  select one edge from the edge set E_C;
  suppose $n(Q_i)n(Q_j)^m$ is the selected edge; $\{n(Q_i)n(Q_j)^m$ is used instead of $n(Q_i)n(Q_j)$ in order to distinguish among multiple edges between $n(Q_i)$ and $n(Q_j)\}$
  $e_{i,j} \leftarrow$ the common boundary edge shared by $\partial V(Q_i)$ and $\partial V(Q_j)$ corresponding to $n(Q_i)n(Q_j)^m$;
  if $e_{i,j}$ meets an unblocked cycle $C \subset V$, which encloses both Q_i and Q_j **then**
    guard the point where $e_{i,j}$ meets C;
  else
    choose one of two end points of $e_{i,j}$ and guard the point;
  end if
  mark all cycles containing the guarded node as blocked;
  selected edge $n(Q_i)n(Q_j)^m$ is removed from the edge set E_C;
until there is no edge left in E_C;

Proof. We already know that $g_I(V) \leq$MinGuard(V) using Lemma 7. If MinGuard$(V) \leq |\alpha(V^*)|$, then $g_I(V) \leq$MinGuard$(V) \leq |\alpha(V^*)|$. This further implies that $g_I(V) \leq |\alpha(V^*)|$.

It remains to show that MinGuard$(V) \leq |\alpha(V^*)|$. We prove by contradiction. Suppose that $|\alpha(V^*)| <$MinGuard(V). Since MinGuard(V) is the minimum number of guarded nodes to block all cycles in V, $|\alpha(V^*)| <$MinGuard(V) implies that as we guard $|\alpha(V^*)|$ nodes in V, there is an unblocked cycle, say C, in V.

$V^*[C] \subset V^*$ is connected according to Lemma 2. Since V^* is a tree graph, $V^*[C] \subset V^*$ is also a tree graph. Therefore, $V^*[C] = \partial(V^*[C])$.

We consider two cases depending on the structure of $\partial(V^*[C])$. For these two cases, we will prove that an unblocked cycle, C, cannot exist as we guard $|\alpha(V^*)|$ nodes in V. Algorithm 6 is used to guard $|\alpha(V^*)|$ nodes in V. That C cannot exist as we guard $|\alpha(V^*)|$ nodes in V further implies that MinGuard$(V) \leq |\alpha(V^*)|$.

1. $\partial(V^*[C])$ is a node. In this case, C contains only one cell.

2. $\partial(V^*[C])$ is a tree graph containing at least one edge. In this case, C contains

more than one cell.

1. Consider the case where $\partial(V^*[C])$ is a node. This implies that C is a cycle basis containing only one cell.

We guard $|\alpha(V^*)|$ nodes in V using algorithm 6. According to the definition of an edge cover, all nodes in V^* are covered by the edges in $\alpha(V^*)$. Since each node in V^* corresponds to a cycle basis, we block all cycle bases contained in V by guarding $|\alpha(V^*)|$ nodes in V. Therefore, C, which is a cycle basis, is blocked as we guard $|\alpha(V^*)|$ nodes in V.

2. Consider the case where $\partial(V^*[C])$ is a tree graph containing at least one edge. There exists at least one edge, say $n(Q_1)n(Q_2)$, in $\partial(V^*[C])$, whose one end point has degree one. Suppose $n(Q_1)$ has degree one. Since $\partial(V^*[C])$ is a tree graph, $n(Q_1)n(Q_2) \in E(V^*)$ indicates that there is only one common boundary edge shared by $\partial V(Q_1)$ and $\partial V(Q_2)$. Let $e_{1,2}$ denote the common boundary edge between $\partial V(Q_1)$ and $\partial V(Q_2)$.

In the proof of step 1, we proved that all cycle bases in V are blocked by guarding $|\alpha(V^*)|$ nodes in V. Thus, there is a guarded node in a cycle basis representing $\partial V(Q_1)$. This guarded node can exists on $e_{1,2}$ or outside $e_{1,2}$. If this guarded node is on $e_{1,2}$, then it is at one end point of $e_{1,2}$ using algorithm 6. Otherwise, the guarded node is on C, since $n(Q_1)$ has degree one. In both cases, C is blocked.

Until now, we proved that C is blocked as we guard $|\alpha(V^*)|$ nodes in V using algorithm 6. □

We derive an upper bound for $g_I(V)$ in the case where V^* is a connected graph with more than one node. We need to introduce a new concept for this purpose. If an edge cover of V^* further satisfies that every cycle in V^* contains at least one edge in this edge cover, then we denote the edge cover as a *cycle-blocking edge cover* of V^*. Let $\beta(V^*) \subset E(V^*)$ denote a cycle-blocking edge cover of V^* with the minimum cardinality, i.e., the fewest number of edges.

In Figure 34, V^* is depicted with normal or dotted lines. The edges in an edge cover of V^* are depicted with dotted lines. The left sub-figure depicts a cycle-blocking edge cover of V^*. In the right sub-figure, a cycle consisting of four nodes ($n(O_1)$, $n(O_4)$, $n(O_6)$, and $n(O_2)$) does not contain a dotted edge. Thus, the right sub-figure shows an edge cover of V^*, which is not a cycle-blocking edge cover.

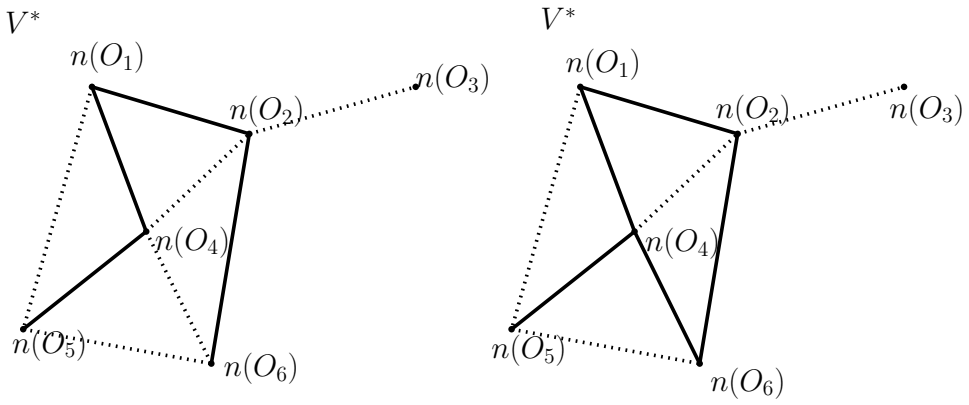


Figure 34: V^* depicted with normal or dotted lines. The left sub-figure depicts a cycle-blocking edge cover of V^*. The right sub-figure shows an edge cover of V^*, which is not a cycle-blocking edge cover.

In the case where V^* is a connected graph with more than one node, we replace $\alpha(V^*)$ in algorithm 6 by $\beta(V^*)$ to guard $|\beta(V^*)|$ nodes in V. In the following theorem, we prove that guarding $|\beta(V^*)|$ nodes using algorithm 6 blocks all cycles in V.

Theorem 9. *For the Voronoi diagram $V = (N(V), E(V))$ such that V^* is a connected graph with more than one node, $g_I(V) \leq |\beta(V^*)|$.*

Proof. Since $g_I(V) \leq$MinGuard(V) using Lemma 7, we will prove that MinGuard(V) $\leq |\beta(V^*)|$ similar to the proof of Theorem 8.
By contradiction, suppose that $|\beta(V^*)| <$MinGuard(V). $|\beta(V^*)| <$MinGuard(V) implies that as we guard $|\beta(V^*)|$ nodes in V, there is an unblocked cycle, say C, in V.

Since $V^*[C]$ is connected according to Lemma 2, any two nodes in $\partial(V^*[C]) \subset V^*[C]$ are connected by a path in $\partial(V^*[C])$. This implies that $\partial(V^*[C])$ is connected.

Since $\partial(V^*[C])$ is connected, we consider two cases depending on the structure of $\partial(V^*[C])$. For these two cases, we will prove that an unblocked cycle, C, cannot exist as we guard $|\beta(V^*)|$ nodes in V, i.e., MinGuard$(V) \leq |\beta(V^*)|$. Algorithm 6 using $\beta(V^*)$ is applied to guard $|\beta(V^*)|$ nodes in V.

1. $\partial(V^*[C])$ is a tree graph.

2. $\partial(V^*[C])$ contains a cycle.

1. Since $\partial(V^*[C])$ is a tree graph, $\partial(V^*[C])$ can be a node or a tree graph containing at least one edge. Note that a cycle-blocking edge cover satisfies the condition for an edge cover. Hence, we replace $\alpha(V^*)$ in the proof of step 1 and 2 in Theorem 8 by $\beta(V^*)$ to obtain the result that C is blocked as we guard $|\beta(V^*)|$ nodes in V.
2. Consider the case where $\partial(V^*[C])$ contains a cycle, say C'. Suppose $N(C') = \{n(Q_1), n(Q_2), ..n(Q_{|N(C')|})\}$ and $E(C') = \bigcup_{i \leq |N(C')|}\{n(Q_i)n(Q_{i+1})\}$. In Figure 35, $C' \subset \partial(V^*[C])$ is depicted with dashed line segments on $\partial(V^*[C])$.

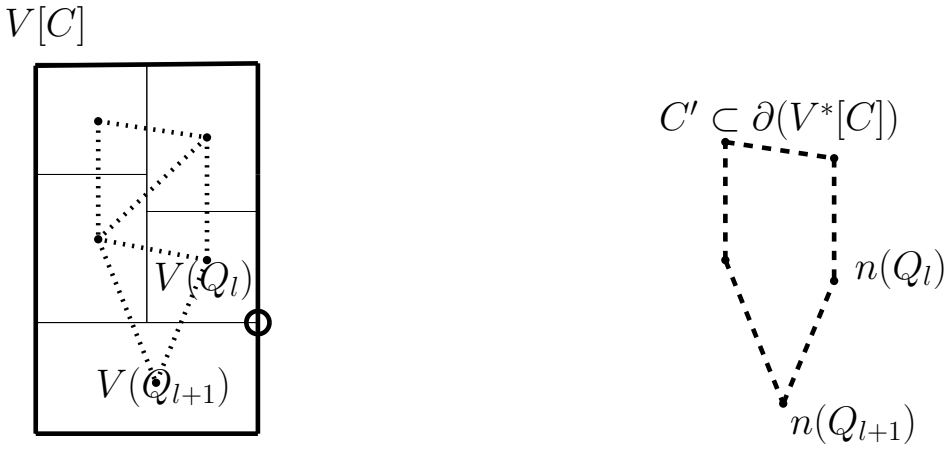


Figure 35: C, $V[C]$, $V^*[C]$, and $\partial(V^*[C])$ are identical to those in Figure 30. $C' \subset \partial(V^*[C])$ is depicted with dashed line segments on $\partial(V^*[C])$.

According to the definition of a cycle-blocking edge cover, $C' \subset V^*$ must contain an edge, say $n(Q_l)n(Q_{l+1})^m$ where m is used to distinguish among multiple edges between $n(Q_l)$ and $n(Q_{l+1})$, in $\beta(V^*) \subset E(V^*)$. Similar to $\alpha(V^*)$, there is one

common boundary edge, say $e_{l,l+1}$, shared by $\partial V(Q_l)$ and $\partial V(Q_{l+1})$ corresponding to $n(Q_l)n(Q_{l+1})^m$. Since both $n(Q_l)$ and $n(Q_{l+1})$ are on the boundary of the unbounded face of $V^*[C]$, $e_{l,l+1}$ meets C. Furthermore, C encloses both Q_l and Q_{l+1}.

Since $n(Q_l)n(Q_{l+1})^m$ is in $\beta(V^*)$, $n(Q_l)n(Q_{l+1})^m$ will be selected while running algorithm 6. Note that C is unblocked and encloses both Q_l and Q_{l+1}. Therefore, we guard the node where $e_{l,l+1}$ meets C according to algorithm 6. In Figure 35, the circle on C represents the guarded node where $e_{l,l+1}$ meets C. In this way, C is blocked as we guard $|\beta(V^*)|$ nodes in V.

□

Lastly, we will derive a relation between $g_I(V)$ and the number of obstacles in the workspace.

Corollary 3. *For the Voronoi diagram* $V = (N(V), E(V))$, $g_I(V) \leq M - 1$.

Proof. In Lemma 2, $V^* = V^*[C]$ if we choose $\partial V(O_M)$ as C. According to Lemma 2, V^* is a connected graph. Depending on the structure of V, V^* can be one node or a connected graph with more than one node.

First, we consider the case where V^* is one node. This case indicates that there is only one cycle in V. In Lemma 8, we derived that $g_I(V) = 1$. Since $M = 2$ for this case, we have $g_I(V) \leq M - 1$.

Next, consider the case where V^* is a connected graph with more than one node. We can choose the edge cover $E_C(V^*)$ as $E(V^*)$. $E(V^*)$ is a cycle-blocking edge cover, since every cycle in V^* contains at least one edge in $E(V^*)$. Moreover, we get

$$|\beta(V^*)| \leq |E(V^*)|, \tag{96}$$

since $\beta(V^*)$ is a cycle-blocking edge cover with the minimum cardinality.

Since V^* is connected, we obtain

$$|E(V^*)| \leq |N(V^*)|. \tag{97}$$

Theorem 9 shows that $g_I(V) \leq |\beta(V^*)|$ in the case where V^* is a connected graph with more than one node. Using (97) and (96), we get $g_I(V) \leq |\beta(V^*)| \leq |N(V^*)| = M - 1$. This further implies that $g_I(V) \leq M - 1$. □

Note that even though the SCENT algorithms produce a communication infrastructure defined by the information network, mobility is restricted to the underlying Voronoi diagram. As such, Theorem 8, Theorem 9, and Corollary 3 do describe bounds on the number of guards needed when the graph structures (the Voronoi diagram and the information network) are obtained from the SCENT algorithms.

4.4 Simulation and Experimental Results

In this section, the time efficiency of the SCENT algorithms is demonstrated in MATLAB simulations, as well as in experimental results using two mobile robots.

4.4.1 MATLAB Simulation Results

The SCENT algorithms are theoretically sound, but we can improve the time efficiency of the algorithms without violating the correctness of the algorithms. One major idea in our algorithms is to expand the enclosure for each vehicle while avoiding overlapping with the enclosures built by other vehicles. Therefore, we can improve the time efficiency of the SCENT algorithms by speeding up the expansion of the enclosure for every vehicle. Two strategies in Section 3.6 can be utilized to speed up the expansion of the enclosure for every vehicle.

Figure 28 (see Section 3.6) shows one vehicle constructing the Voronoi diagram in a rectangular shaped workspace using the BE algorithms with improvement over the time efficiency. Recall that the total exploration time in Figure 28 is 36.3 time units.

Figure 36 shows two vehicles constructing the Voronoi diagram in the workspace identical to the workspace in Figure 28 using the improved SCENT algorithms. In Figure 36, two strategies in Section 3.6 are utilized to speed up the expansion of the

enclosure for every vehicle. The initial positions of the vehicles are $(2, 20)$ and $(45, 2)$ respectively. The obstacle boundaries are shown in thick red curves. Moreover, the trajectories of the vehicles are marked with points and circles respectively. Along the trajectory of each vehicle, intersections are marked with large green dots. The exploration time using two vehicles is 17.35 time units, which is less than half of the exploration time using one vehicle. This simulation result verifies that under the improved SCENT algorithms, we can enhance the time efficiency of the exploration considerably as the number of vehicles increases.

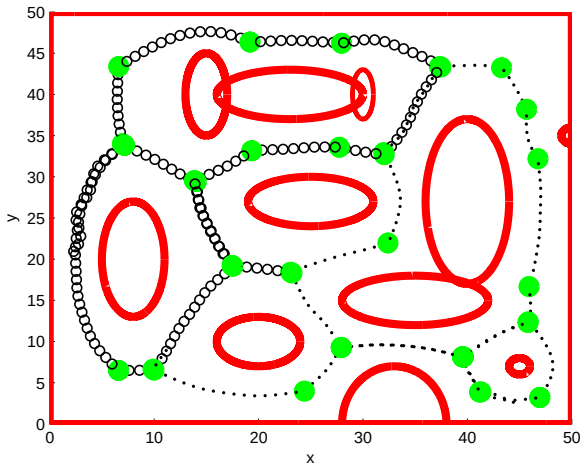


Figure 36: Two vehicles constructing the Voronoi diagram using the improved SCENT algorithms.

4.4.2 Experimental Results

We verify the SCENT algorithms through experiments using two Khepera III robots [72]. Each robot has nine IR sensors and five sonar sensors. In the experiments, we use only IR sensors for range measurements, and each robot localizes itself based on an odometry system.

The experiments are performed in a small bounded workspace with four obstacles. In this small scale environment, the communication range of a robot can cover the entire workspace. Hence, one robot can communicate with another robot directly

using the built-in communication module. This implies that the two robots do not need to deploy information nodes in the workspace.

Since robots do not deploy information nodes in the experiments, we need to simulate what happens when a robot meets a deployed information node. To simulate this, a camera, which is connected to a PC, is mounted on top of the workspace. This is depicted in Figure 37. When a robot meets an intersection, it sends a triggering signal to the PC so that the camera captures the position of the robot in the workspace. Based on the captured image, the PC calculates the coordinate of the intersection. The PC further generates a virtual information network built by the two robots. Since we simulate that a robot deploys an information node whenever it meets an intersection, each node in the virtual information network corresponds to an intersection in the obstacle environment[2].

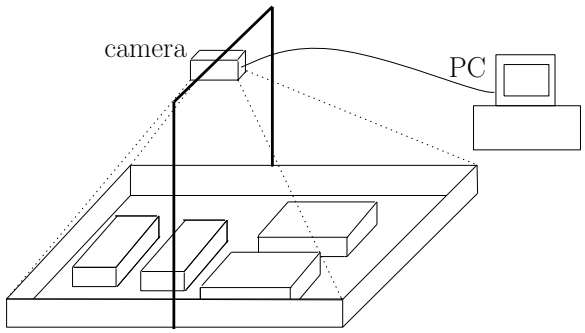


Figure 37: The illustration of the experimental environment.

Successful experimental results of the SCENT algorithms are depicted in Figure 38. The two sub-figures in the top row show the obstacle environment detected using IR sensors for each robot. The trajectories of the robots are depicted as blue curves in between obstacles, and intersections are marked with green rectangles. Furthermore, as the two robots move around, a virtual information network is constructed as shown in the bottom right sub-figure.

[2]Due to sensor measurement noise, a robot may not detect an intersection at the correct intersection point. Hence, when the relative distance between two intersections is less than a certain threshold, we regard the two intersections as the same node in the virtual information network.

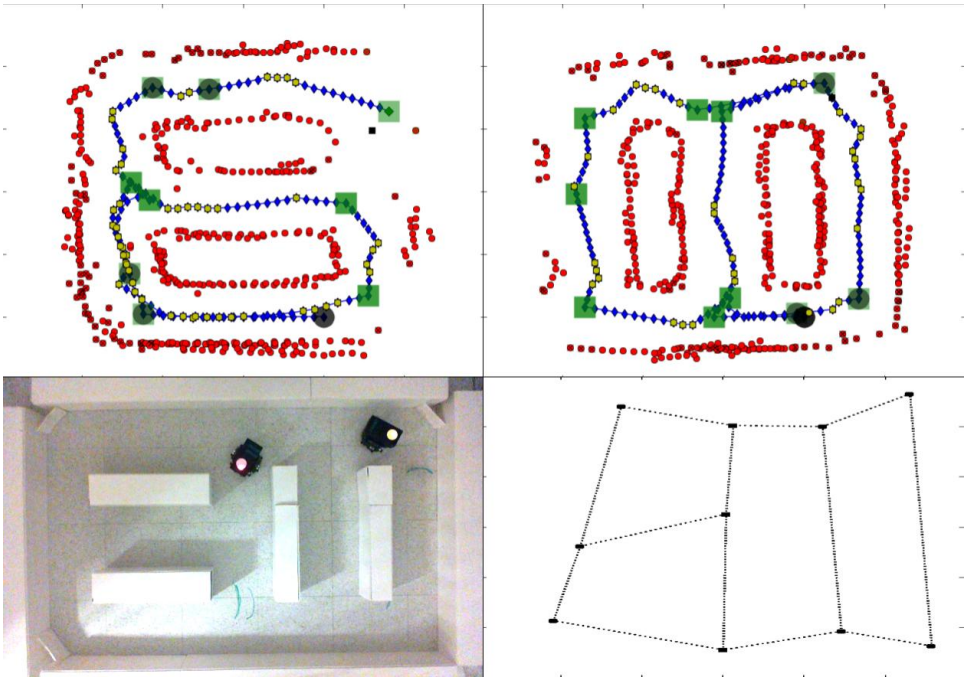


Figure 38: Experimental results of the SCENT algorithms. Two sub-figures in the top row display the obstacle environment detected using IR sensors for each robot. The bottom right sub-figure shows a virtual information network generated in real time.

4.5 Conclusions

We develop the SCENT algorithms for multiple vehicles. The theme of the SCENT algorithms is that each vehicle explores its local area by incrementally expanding the already visited parts of the environment. At the same time, every vehicle deploys information nodes at selected locations and, as a result, an information network is created concurrently with a topological map based on Voronoi diagrams.

We prove that such a cooperative strategy leads to time-efficient construction of the Voronoi diagram in an unknown environment. In addition, we verify the efficiency of the SCENT algorithms through MATLAB simulations as well as experiments using two mobile robots.

Once the information network is completely constructed using the SCENT algorithms, this network can then be utilized as a basis for capturing intruders in the workspace.

CHAPTER V

CONCLUSIONS AND FUTURE DIRECTIONS

5.1 Conclusions

The main contributions of this thesis are summarized as follows:

- *curve-tracking control of one vehicle with range sensors.* We introduce curve-tracking control which only requires two narrow aperture range sensors, pointing to a fixed direction relative to the moving direction of the vehicle. Under such a sensor configuration, singularities are bound to occur in the control laws when tracking concave curves. In order to overcome these singularities, we derive a hybrid strategy of switching between control laws when the vehicle gets close to singularities.

- *exploration strategy of one vehicle with range sensors.* We present the boundary expansion (BE) algorithms and the tracking control law that enable the construction of the Voronoi diagram in an initially unknown environment using a single vehicle equipped with range sensors. The BE algorithms are provably complete, and the convergence of the tracking control law is guaranteed. We implement the algorithms and the tracking control law using a miniature robot localizing itself based on an odometry system. The robot uses only Infrared(IR) sensors for range measurements.

- *multi-robot exploration and mapping.* The SCENT algorithms extend the BE algorithms to multiple vehicles. Each vehicle explores its local area by incrementally expanding the already visited parts of the environment. At the same time, every vehicle deploys information nodes at selected locations and, as a

result, an information network is created concurrently with a topological map. A performance analysis of the SCENT algorithms verifies that in a bounded workspace, the time spent to complete the exploration decreases as the number of vehicles increases. Furthermore, simulation and experimental results are presented to demonstrate the effectiveness of the SCENT algorithms.

- *capturing intruders on a graph.* Once the information network is completely constructed using the SCENT algorithms, this network can be used as a basis for capturing intruders in the workspace. We consider a simplified scenario such that searchers and intruders move along edges of the Voronoi diagram. We derive theoretical upper bound for the minimum number of searchers required to capture all intruders on a general graph, which leads to a result on the Voronoi diagram.

These technical contributions are supported by the following publications: [53, 54, 52, 51, 50, 48].

5.2 Future Directions

This section provides future directions of the research presented in this thesis.

5.2.1 Future Directions of Tracking Control

Tracking control can be applied for controlling any type of autonomous vehicle so that the vehicle follows curbs or lane markings. Several improvements of the curve-tracking control in Chapter 2 can be expected.

Since we have derived the tracking model for mounting angle α in Chapter 2, an extension for the controller from $\alpha = \pi/2$ to the general case should be possible. In Chapter 2, we estimated the curvature of the curve at the detected point based on range sensor measurements. We observed from simulation that such an estimate contains noise that may cause unnecessary switching, which affects tracking performance.

Hence, a filtering algorithm for curvature estimation can be developed to reduce noise. In addition, multiple vehicles can be coordinated, similar to [90, 21, 34], for dynamic boundary estimation.

We can extend tracking control into 3D environments. In exploration of 3D scalar fields using multiple sensor platforms, noise in measurements of the scalar fields should be considered. We can adjust the formation of multiple sensors to reduce the effect of noise in measurements of the scalar fields.

5.2.2 Future Directions of the SCENT Algorithms

5.2.2.1 Platform Design for the SCENT Algorithms

To implement the SCENT algorithms in a real application scenario, we consider building hardware platforms for the SCENT algorithms. We require an autonomous robot that deploys information nodes at selected locations. Information nodes should satisfy the condition that the information network is in place once the information nodes are deployed. In a real application scenario, a tiny information node may be desirable to increase the number of nodes that a robot can carry on. In Chapter 4, we assume that a searcher can obtain the position of an intruder using the information network. To implement this task, a deployed information node must have the ability to detect nearby intruders. We need to design information nodes which satisfy these conditions and program each device to make it function as required in the SCENT algorithms.

5.2.2.2 Utilization of the Completely Constructed Information Network

Suppose that the information network is completely constructed using the SCENT algorithms. Then, this network can be utilized in solving coordinated multi-robot tasks. This thesis presents the case where the information network is utilized for capturing intruders. In addition, we can utilize the information network for traffic control. The problem that we ask is as follows: "given a team of agents utilizing the

information network, can we automatically generate provably correct local control strategies from global task specifications given?" One such task specifications can be as follows: each robot goes to its goal as quickly as possible while avoiding collision with one another.

5.2.2.3 2.5D SCENT Algorithms

We can extend the SCENT algorithms from ground to aerial vehicles. In computer graphics and video game development, a 3D scene is often created using a 2.5D strategy in the sense that the movements of video game agents are confined within 2D planes. This strategy reduces the possible outcomes associated with agent movements and the amount of computation needed to render real-time 3D visualization.

As such, we can develop 2.5D SCENT algorithms so that multiple aerial vehicles can explore 2.5D workspace while building a 3D information network. A 2.5D workspace is composed of several 2D planes with different elevations. To explore each plane, the SCENT algorithms in Chapter 4 can be applied.

Once an information network is built as a result of 2.5D SCENT algorithms, then we can utilize the network to solve coordinated multi-robot tasks, such as capturing intruders in 2.5D environments.

APPENDIX A

CONDITIONS FOR A WORKSPACE TO HAVE HEXAGONAL VORONOI CELLS

We give a set of sufficient conditions for a workspace to have identical hexagonal Voronoi cells. Beforehand, we introduce the following concepts: $L(a,b) \subset R^2$ denotes the straight line segment connecting two points a and b in R^2. The angle formed by $L(a,b)$ and $L(a,c)$ is $\angle(b,a,c)$ where $\angle(b,a,c) = \angle(c,a,b)$. $\triangle\ (b,a,c) = L(a,b) \bigcup L(b,c) \bigcup L(c,a) \subset R^2$ denotes the triangle with three vertices a, b, and c.

Furthermore, we need to introduce a *partition point* which partitions an intersection circle into sectors. A partition point corresponds to a closest point on an intersection circle (See Section 3.3.1 for the definition of a closest point) except for the special case where there are infinite number of closest points on an intersection circle. The special case is given as follows. Suppose that a connected segment of an obstacle boundary is on an arc of an intersection circle. In this case, all points on the connected segment correspond to infinite number of closest points. However, among infinite number of closest points on the connected segment, two end points of this connected segment are chosen as two partition points on this intersection circle.

In Figure 39, we illustrate the case where each intersection circle has three partition points and infinite number of closest points. In this figure, one circular obstacle O_i is enclosed by O_M. Obstacle boundaries are plotted as bold curves. See that six intersection circles are tangential to O_i. Each intersection circle has three partition points, and these partition points form three sectors with identical central angle $2\pi/3$.

Lemma 9. *Every obstacle, except for O_M, in W has a hexagonal Voronoi cell if the following conditions are satisfied:*

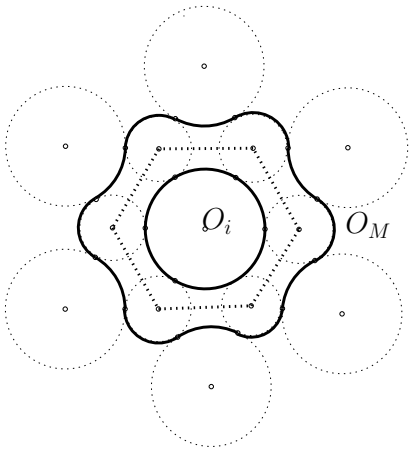


Figure 39: Illustration of one obstacle O_i enclosed by O_M. Obstacle boundaries are plotted as bold curves. To make a hexagonal Voronoi cell, one obstacle O_i has a circular shape and O_M has a symmetric shape surrounding O_i.

1. *each obstacle, except for O_M, has an identical circular shape.*

2. *for every intersection in W, the intersection circle has three partition points with identical radius. On each intersection circle, three partition points form three sectors with identical central angle $2\pi/3$.*

Proof. We organize our proof in two steps:

1. show that every Voronoi edge on $\partial V(O_i)$, where $i \neq M$, is a straight line with identical length.

2. show that the number of intersections on $\partial V(O_i)$, where $i \neq M$, is 6.

1. We prove that every Voronoi edge on $\partial V(O_i)$ where $i \neq M$ is a straight line of identical length. We label the intersections on $\partial V(O_i)$ in the counterclockwise direction as $P_{i,1}, P_{i,2}, \ldots P_{i,N}$ where N is the number of intersections on $\partial V(O_i)$.

Consider a Voronoi edge connecting two points $P_{i,n}$ and $P_{i,n+1}$. We prove that the Voronoi edge connecting $P_{i,n}$ and $P_{i,n+1}$ is $L(P_{i,n}, P_{i,n+1})$. Suppose that the Voronoi edge is shared by $V(O_i)$ and $V(O_k)$ where $k \neq i$. See Figure 40 for illustration. In this figure, circular obstacles O_i and O_k have the identical radius R. Let C_i and C_k denote the center of O_i and O_k respectively.

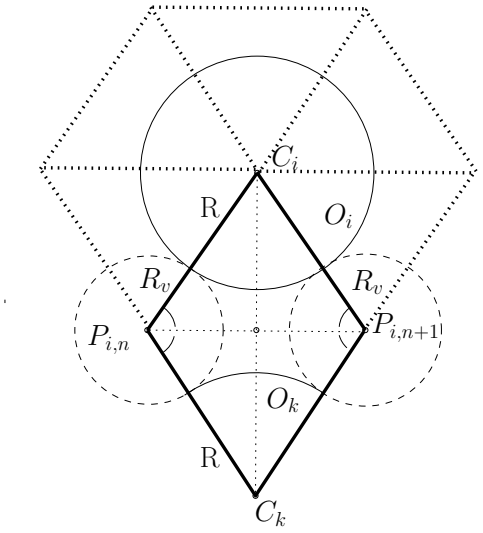


Figure 40: Circular obstacles O_i and O_k have the identical radius R. An intersection circle, whose radius is R_v, is depicted with a dashed circle.

If $L(P_{i,n}, P_{i,n+1})$ passes through the center of $L(C_i, C_k)$ normal to $L(C_i, C_k)$, then the points on $L(P_{i,n}, P_{i,n+1})$ are equidistant from the two points C_i and C_k. Furthermore, the points that are equidistant from C_i and C_k are equidistant from O_i and O_k, since both O_i and O_k have the identical radius R. In short, if $L(P_{i,n}, P_{i,n+1})$ passes through the center of $L(C_i, C_k)$ normal to $L(C_i, C_k)$, then the points on $L(P_{i,n}, P_{i,n+1})$ are equidistant from two obstacles O_i and O_k, i.e., the Voronoi edge connecting $P_{i,n}$ and $P_{i,n+1}$ is $L(P_{i,n}, P_{i,n+1})$.

It remains to prove that $L(P_{i,n}, P_{i,n+1})$ passes through the center of $L(C_i, C_k)$ normal to $L(C_i, C_k)$. Since a circular obstacle and an intersection circle are tangential to each other, we can draw $L(P_{i,n}, C_i)$ such that one partition point for $P_{i,n}$ is on $L(P_{i,n}, C_i)$, as illustrated in Figure 40. Similarly, we can draw three line segments: $L(P_{i,n}, C_k)$, $L(P_{i,n+1}, C_i)$, and $L(P_{i,n+1}, C_k)$. The length of each line segment is $R+R_v$ where R_v is the radius of an intersection circle. In this way, we draw a rhombus, a quadrilateral whose four edges all have identical length $R + R_v$, with four vertices $P_{i,n}, C_i, P_{i,n+1}, C_k$. The rhombus is plotted with bold line segments in Figure 40. Using the properties of a rhombus, we get $\angle(C_k, P_{i,n}, P_{i,n+1}) = \angle(C_i, P_{i,n+1}, P_{i,n}) =$

$\angle(C_i, P_{i,n}, P_{i,n+1}) = \angle(C_k, P_{i,n+1}, P_{i,n})$. Also, $\angle(C_k, P_{i,n}, C_i) = \angle(C_i, P_{i,n+1}, C_k) = 2\pi/3$, since three partition points of $P_{i,n}$ (or $P_{i,n+1}$) form three sectors with identical central angle $2\pi/3$. Hence, we get $\angle(C_k, P_{i,n}, P_{i,n+1}) = \angle(C_i, P_{i,n}, P_{i,n+1}) = \pi/3$, which implies that both $\triangle\ (C_k, P_{i,n+1}, P_{i,n})$ and $\triangle\ (C_i, P_{i,n+1}, P_{i,n})$ are equilateral triangles. Since $\triangle\ (C_k, P_{i,n+1}, P_{i,n})$ and $\triangle\ (C_i, P_{i,n+1}, P_{i,n})$ are equilateral triangles, $L(P_{i,n}, P_{i,n+1})$ passes through the center of $L(C_i, C_k)$ normal to $L(C_i, C_k)$.

2. We prove that the number of intersections on $\partial V(O_i)$, where $i \neq M$, is 6. For all $l \leq N$, $\angle(P_{i,l}, C_i, P_{i,l+1}) = \pi/3$, since $\triangle\ (C_i, P_{i,l+1}, P_{i,l})$ is an equilateral triangle. Notice that $\frac{2\pi}{\pi/3} = 6$. Hence, there exist 6 intersections on $\partial V(O_i)$ where $i \neq M$.

□

REFERENCES

[1] Andersson, S. B. and Park, J., "Tip steering for fast imaging in AFM," in *proc. of American Control Conference*, (Portland, OR, USA), pp. 2469–2474, 2005.

[2] Aurenhammer, F., "Voronoi diagrams-a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, pp. 345–405, 1991.

[3] Bailey, T. and Durrant-Whyte, H., "Simultaneous localization and mapping: Part II," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 108–117, 2006.

[4] Barriére, L., Ere, L. B., Flocchini, P., Fraigniaud, P., and Santoro, N., "Capture of an intruder by mobile agents," in *proc. of the fourteenth annual ACM symposium on Parallel algorithms and architectures*, (Canada), pp. 200–209, 2002.

[5] Bhattacharya, P. and Gavrilova, M. L., "Roadmap-based path planning - using the Voronoi diagram for a clearance-based shortest path," *IEEE Robotics and Automation Magazine*, vol. 15, pp. 58–66, 2008.

[6] Bienstock, D. and Seymour, P., "Monotonicity in graph searching," *Journal of Algorithms*, vol. 12, no. 2, pp. 239–245, 1991.

[7] Branicky, M. S., "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43(4), pp. 475–482, 1998.

[8] Brooks, R. A., *Cambrian Intelligence: The Early History of the New AI*. MIT Press, 1999.

[9] Brown, G., "Point density in stems per acre," *Newzealand Forestry Service Research Notes*, vol. 38, pp. 1–11, 1965.

[10] Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S., "Collaborative multi-robot exploration," in *proc. of IEEE International Conference on Robotics and Automation*, (CA, USA), pp. 476–481, 2000.

[11] Burgard, W., Moors, M., Stachniss, C., and Schneider, F., "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, pp. 376–386, 2005.

[12] Calabi, E., Olver, P. J., Shakiban, C., Tannenbaum, A., and S.Haker, "differential and numerically invariant signature curves applied to object recognition," *International Journal of Computer Vision*, vol. 26, pp. 107–135, 1998.

[13] Carmo, M. D., *Differential Geometry of Curves and Surfaces*. Prentice Hall, 1976.

[14] Choset, H. and Burdick, J., "Sensor-based exploration: The hierarchical generalized Voronoi diagram," *The International Journal of Robotics Research*, vol. 19, pp. 96–125, 2000.

[15] Choset, H., Konukseven, I., and Burdick, J., "Mobile robot navigation: issues in implementating the generalized Voronoi graph in the plane," in *proc. of IEEE/SICE/RSJ International Conference on Multisensor Fusion and Integration for Intelligent Systems*, (Washington DC, USA), pp. 241–248, 1996.

[16] Choset, H., Konukseven, I., and Rizzi, A., "Sensor based planning: a control law for generating the generalized Voronoi graph," in *proc. of 8th International Conference on Advanced Robotics*, (CA, USA), pp. 333–338, 1997.

[17] Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L., and Thrun, S., *Principles of robot motion*. MIT Press, 2005.

[18] Choset, H. and Nagatani, K., "Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 125–137, 2001.

[19] Choset, H., Walker, S., Eiamsa-Ard, K., and Burdick, J., "Incremental construction of the hierarchical generalized Voronoi graph," *The International Journal of Robotics Research*, vol. 19, pp. 126–148, 2000.

[20] Choset, H., Walker, S., Eiamsa-Ard, K., and Burdick, J., "Sensor-based exploration: Incremental construction of the hierarchical generalized Voronoi graph," *The International Journal of Robotics Research*, vol. 19(2), pp. 126–148, 2000.

[21] Clark, J. and Fierro, R., "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *proc. of American Control Conference*, (Portland, OR, USA), pp. 3500–3505, 2005.

[22] Corke, P., Detweiler, C., Dunbabin, M., Hamilton, M., Rus, D., and Vasilescu, I., "Experiments with underwater robot localization and tracking," in *proc. of IEEE International Conference on Robotics and Automation*, (Roma, ITALY), pp. 4556–4561, 2007.

[23] Cortés, J., Martínez, S., Karatas, T., and Bullo, F., "Coverage control for mobile sensing networks," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 2, pp. 243–255, 2004.

[24] Culler, D., Estrin, D., and Srivastava, M., "Overview of sensor networks," *IEEE Computer Magazine*, vol. 37, no. 8, pp. 41–49, 2004.

[25] de Verdire, Y. C., "Multiplicities of eigenvalues and tree-width of graphs," *Journal of Combinatorial Theory, Series B*, vol. 74, no. 2, pp. 121–146, 1998.

[26] DeCarlo, R., Branicky, M. S., Pettersson, S., and Lennartson, B., "Perspectives and results on the stability and stabilizability of hybrid systems," *Proc. of the IEEE*, vol. 88(2), pp. 1069–1082, 2000.

[27] Dendris, N. D., Kirousis, L. M., and Thilikos, D. M., "Fugutive-search games on graphs and related parameters," *Theoretical Computer Science*, vol. 172, pp. 233–254, 1997.

[28] Dickmanns, E. D. and Graefe, V., "Applications of dynamic monocular machine vision," *Machine Vision and Applications*, vol. 1, pp. 241–261, 1988.

[29] Dickmanns, E. D. and Mysliwetz, B. D., "Recursive 3-d road and relative ego-state estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, pp. 199–213, 1992.

[30] Douglas, B. W., *Introduction to Graph Theory.* Illinois, USA: Prentice Hall, 2 ed., 2001.

[31] Du, Q., Faber, V., and Gunzburger, M., "Centroidal Voronoi tessellations: Applications and algorithms," *Society for Industrial and Applied Mathematics*, vol. 41, pp. 637–676, 1999.

[32] Durrant-Whyte, H. and Bailey, T., "Simultaneous localization and mapping: Part I," *IEEE Robotics and Automation Magazine*, vol. 13, pp. 99–108, 2006.

[33] Dutta, H., "Survey of approximation algorithms for set cover problem," Master's thesis, University of North Texas, 2009.

[34] Duttagupta, S., Ramamritham, K., and Ramanathan, P., "Distributed boundary estimation using sensor networks," in *proc. of IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, (Vancouver, Canada), pp. 316–325, 2006.

[35] Egerstedt, M., Hu, X., and Stotsky, A., "Control of mobile platforms using a virtual vehicle approach," *IEEE Transactions on Automatic Control*, vol. 46, pp. 1777–1782, 2001.

[36] Fomin, F. V., Fraigniaud, P., and Nisse, N., "Nondeterministic graph searching: From pathwidth to treewidth," *Algorithmica*, vol. 53(3), pp. 358–373, 2009.

[37] Fomin, F. V. and Thilikos, D. M., "An annotated bibliography on guaranteed graph searching," *Theoretical Computer Science*, vol. 399, pp. 236–245, 2008.

[38] Fortune, S. J., "A sweepline algorithm for Voronoi diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987.

[39] Fox, D., Ko, J., Stewart, B., Konolige, K., and Limketkai, B., "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94(7), pp. 1325–1339, 2006.

[40] Fraigniaud, P. and Nisse, N., "Connected treewidth and connected graph searching," *Lecture Notes in Computer Science*, vol. 3887, pp. 479–490, 2006.

[41] Frezza, R. and Picci, G., "On line path following by recursive spline updating," in *proc. of 34th IEEE Conference on Decision and Control*, (New Orleans, LA, USA), pp. 4047–4052, 1995.

[42] Ghrist, R., Lipsky, D., Poduri, S., and Sukhatme, G., "Surrounding nodes in coordinate-free networks," *Algorithmic Foundation of Robotics VII: Springer Tracts in Advanced Robotics*, vol. 47, pp. 409–424, 2008.

[43] Hespanha, J. P. and Morse, A. S., "Stability of switched systems with average dwell-time," in *proc. of 38th IEEE Conference on Decision and Control*, (Phoenix, AZ, USA), pp. 2655–2660, 1999.

[44] Howard, A., "Multi-robot simultaneous localization and mapping using particle filters," *The International Journal of Robotics Research*, vol. 25, no. 12, pp. 1243–1256, 2006.

[45] Justh, E. W. and Krishnaprasad, P. S., "Natural frames and interacting particles in three dimensions," in *proc. of 44th IEEE Conference on Decision and Control*, (Seville, Spain), pp. 2841–2846, 2005.

[46] Justh, E. W. and Krishnaprasad, P. S., "Steering laws for motion camouflage," *Royal Society of London Proceedings Series A*, vol. 462, pp. 3629–3643, 2006.

[47] Khalil, H. K., *Nonlinear Systems*. New Jersey: Prentice Hall, 3 ed., 2002.

[48] Kim, J., Maxon, S., Egerstedt, M., and Zhang, F., "Intruder capturing on a topological map assisted by information networks," in *proc. of IEEE Conference on Decision and Control (submitted)*, (Orlando, USA), 2011.

[49] Kim, J., Maxon, S., Zhang, F., and Egerstedt, M., "Intruder capture beta 1.2." Website, 2010. `http://lamon.gtsav.gatech.edu/~smaxon3/intruder_applet.html`.

[50] Kim, J., Zhang, F., and Egerstedt., M., "Curve tracking control for autonomous vehicles with rigidly mounted range sensors," in *proc. of IEEE Conference on Decision and Control*, (Mexico), pp. 5036–5041, 2008.

[51] Kim, J., Zhang, F., and Egerstedt, M., "Curve tracking control for autonomous vehicles with rigidly mounted range sensors," *Journal of Intelligent and Robotic Systems*, vol. 56, pp. 177–198, 2009.

[52] Kim, J., Zhang, F., and Egerstedt, M., "An exploration strategy based on construction of voronoi diagrams with provable completeness," in *proc. of IEEE Conference on decision and control*, (Shanghai, China), pp. 7024–7029, 2009.

[53] Kim, J., Zhang, F., and Egerstedt, M., "Simultaneous cooperative exploration and networking based on Voronoi diagrams," in *proc. of IFAC Workshop on Networked Robotics*, (Colorado, USA), pp. 1–6, 2009.

[54] Kim, J., Zhang, F., and Egerstedt, M., "A provably complete exploration strategy by constructing Voronoi diagrams," *Autonomous Robots*, vol. 29(3-4), pp. 367–380, 2010.

[55] Kirousis, L. M. and Papadimitriou, C. H., "Interval graphs and searching," *Discrete Mathematics*, vol. 55, no. 2, pp. 181–184, 1985.

[56] Kirousis, L. M. and Papadimitriou, C. H., "Searching and pebbling," *Theoretical Computer Science*, vol. 42, no. 2, pp. 205–218, 1986.

[57] Klein, R., "Abstract Voronoi diagrams and their applications," *Computational Geometry and its Applications*, vol. 333, pp. 148–157, 1988.

[58] Klein, R., *Concrete and Abstract Voronoi diagrams*. Springer, 1990.

[59] Kolling, A. and Carpin, S., "The graph-clear problem: definition, theoretical properties and its connections to multirobot aided surveillance," in *proc. of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Sandiego, USA), pp. 1003–1008, 2009.

[60] Lapaugh, A., "Recontamination does not help to search a graph," *Journal of the ACM*, vol. 40(2), pp. 224–245, 1993.

[61] Lavalle, S. M., *Planning Algorithms*. Cambridge University Press, 2006.

[62] Li, K. and Baillieul, J., "Data-rate requirements for nonlinear feedback control," in *proc. of 6th IFAC Symp. Nonlinear Contr. Sys.*, (Stuttgart, Germany), pp. 1277–1282, 2004.

[63] Liberzon, D. and Morse, A. S., "Benchmark problems in stability and design of switched systems," *IEEE Control Systems Magazine*, pp. 59–70, 1999.

[64] Lindeberg, T., "Scale-space for discrete signals," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12(3), pp. 234–254, 1990.

[65] Ma, Y., Koseck'a, J., and Sastry, S., "Vision guided navigation for a nonholonomic mobile robot," in *proc. of 36th IEEE Conference on Decision and Control*, (San Diego, CA, USA), pp. 3069–3074, 1997.

[66] Ma, Y., Kosecka, J., and Sastry, S., "Vision guided navigation for a nonholonomic mobile robot," *IEEE Transactions on Robotics and Automation*, vol. 15(3), pp. 521–536, 1999.

[67] Makedon, F. and Sudborough, I. H., "On minimizing width in linear layouts," *Discrete Applied Mathematics*, vol. 23(3), pp. 243–265, 1989.

[68] Martínez, S., Cortés, J., and Bullo, F., "Motion coordination with distributed information," *IEEE Control Systems Magazine*, vol. 27(4), pp. 75–88, 2007.

[69] Mead, R., "A relation between the individual plant-spacing and yield," *Ann. of Bot., N. S.*, vol. 30, pp. 301–309, 1966.

[70] Megiddo, N., Hakimi, S. L., Garey, M. R., Johnson, D. S., and Papadimitriou, C. H., "The complexity of searching a graph," *Journal of the ACM*, vol. 35, pp. 18–44, 1988.

[71] Micaelli, A. and Samson, C., "Trajectory tracking for unicycle-type and two-steering-wheels mobile robots," tech. rep., INRIA, 1993.

[72] Mondada, F., Franzi, E., and Ienne, P., "Mobile robot miniaturisation: A tool for investigation in control algorithms," in *proc. of the Third International Symposium on Experimental Robotics*, (Kyoto, Japan), pp. 501–513, 1993.

[73] Nagatani, K. and Choset, H., "Toward robust sensor based exploration by constructing reduced generalized Voronoi graph," in *proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Kyongju, Korea), pp. 1687–1692, 1999.

[74] Newman, D., "The hexagon theorem," *IEEE Transactions on Information Theory*, vol. 28, pp. 137–139, 1982.

[75] Parsons, T. D., "Pursuit-evasion in a graph," *Theory and Applications of Graphs, Lecture Notes in Mathematics, Springer-Verlag*, vol. 642, pp. 426–441, 1978.

[76] Rao, N. S. V., Stoltzfus, N., and Iyengar, S. S., "A retraction method for learned navigation in unknown terrains for a circular robot," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 699–707, 1991.

[77] Raviv, D. and Herman, M., "Nonreconstruction approach for road following," in *proc. of SPIE: Intelligent Robots and Computer Vision*, (Boston, USA), pp. 2–12, 1992.

[78] Reddy, P. V., Justh, E. W., and Krishnaprasad, P. S., "Motion camouflage in three dimensions," in *proc. of 45th IEEE Conference on Decision and Control*, (San Diego, CA, USA), pp. 3327–3332, 2006.

[79] Richerby, D. and Thilikos, D. M., "Graph searching in a crime wave," *SIAM J. discrete mathematics*, vol. 23(1), pp. 349–368, 2009.

[80] Samson, C., "Control of chained systems: Application to path-following and time-varying point-stabilization of mobile robots," *IEEE Transactions on Automatic Control*, vol. 40, pp. 64–77, 1995.

[81] Seymour, P. D. and Thomas, R., "Graph searching and a min-max theorem for tree-width," *J. Combin. Theory Ser. B*, vol. 58, pp. 22–33, 1993.

[82] Svec, P., *Using Methods of Computational Geometry in Robotics*. PhD thesis, Brno University of Technology, 2007.

[83] Takahashi, A., Ueno, S., and Kajitani, Y., "Mixed searching and proper-path-width," *Theoretical Computer Science*, vol. 137, no. 2, pp. 253–268, 1995.

[84] Thurn, S. and Burgard, W., *Probabilistic Robotics*. Cambridge, MA: MIT, 2005.

[85] Tomlin, C. and Sastry, S., "Switching through singularities," in *proc. of 36th IEEE Conference on Decision and Control*, (San Diego, CA, USA), pp. 1–6, 1997.

[86] Wein, R., v. d. Berg, J. P., and Halperin, D., "The visibility–voronoi complex and its applications," in *proc. of the Twenty-first Annual Symposium on Computational Geometry*, (Pisa, Italy), pp. 63–72, 2005.

[87] Zhang, F., Fiorelli, E., and Leonard, N. E., "Exploring scalar fields using multiple sensor platforms: Tracking level curves," in *proc. of 46th IEEE Conference on Decision and Control*, (New Orleans, LA, USA), pp. 3579–3584, 2007.

[88] Zhang, F., Fratantoni, D. M., Paley, D., Lund, J., and Leonard, N. E., "Control of coordinated patterns for ocean sampling," *International Journal of Control*, vol. 80, pp. 1186–1199, 2007.

[89] Zhang, F., Justh, E., and Krishnaprasad, P. S., "Boundary following using gyroscopic control," in *proc. of 43rd IEEE Conference on Decision and Control*, (Atlantis, Paradise Island, Bahamas), pp. 5204–5209, 2004.

[90] Zhang, F. and Leonard, N., "Generating contour plots using multiple sensor platforms," in *proc. of IEEE Symposium on Swarm Intelligence*, (Pasadena, California), pp. 309–314, 2005.

[91] Zhang, F. and Leonard, N. E., "Coordinated patterns of unit speed particles on a closed curve," *Systems and Control Letters*, vol. 56, pp. 397–407, 2007.

[92] Zhang., F. and Leonard, N. E., "Cooperative control and filtering for cooperative exploration," *IEEE Transactions on Automatic Control*, vol. 55(3), pp. 650–663, 2010.

[93] Zhang, F., O'Connor, A., Luebke, D., and Krishnaprasad, P. S., "Experimental study of curvature-based control laws for obstacle avoidance," in *proc. of IEEE International Conference on Robotics and Automation*, (New Orleans, LA, USA), pp. 3849–3854, 2004.

VITA

Jonghoek Kim is a graduate research assistant and Ph.D. candidate at the School of Electrical and Computer Engineering in Georgia Institute of Technology. His research focuses on developing motion control, planning, and coordination of robots equipped with range sensors. Jonghoek Kim received his M.S. in Electrical and Computer Engineering from Georgia Institute of Technology in 2008 and his B.S. in Electrical and Computer Engineering from Yonsei university, South Korea in 2006.